1992 / FEBRUÁR

ÁRA: 196 FT

ALAPLAP

MIKROSZÁMÍTÓGÉP MAGAZIN MÁGNESLEMEZ MELLÉKLETTEL



A HÓNAP TÉMÁJA:

DIGITÁLIS BÁBEL

„Perszonális" programozás

Mi fér a memóriába — és hova?

Program kezdő harkályoknak

A MÁGNESLEMEZEN:

Robot Robi
Gépírást oktató demó
Modula példaprogramok
A GEM rendszer XVIII.
Játék: Égitámadás

Koprocesszor-specialitások

DataFlex adatbázis-kezelő

Turbo Pascal — közkincsben

Klappolhat a Clipper

Unix-fájlok, mint fekete dobozok

PC-sakk

Ha a megbízhatóság a döntő...



VIGYÁZAT! Jól bevezetett és hírnévnek örvendő márkanevünkkel kétes minőségű, hasonló hangzású nevek élnek vissza!

A MITAC 17 éves információipari hátterével a technológia egyik távol-keleti vezetője. Igen szigorú minőségbiztosító rendszerének és hatalmas kutató-fejlesztő beruházásainak eredményeképpen termékei a világ 65 országában váltak a korszerűség és a megbízhatóság szinonimájává.

A megbízható gyártó termékei csak megbízható forgalmazó tevékenysége nyomán képesek a felhasználó javát szolgálni.

Ezért esett a MITAC választása hazánkban az INTERAG-ra.

Forgalmazó: INTERAG

INTERAG INFORMATIKA
Budapest 1136 Pannónia u. 11.
Tel./fax.: 132-9375 Molnár Péter

People Committed To InfoTech

MiTAC

INFORMÁCIÓKÉRÉS: 01 ▶

ALAPLAP

Mikroszámítógép magazin mágneslemez melléklettel

Megjelenik havonta

Főszerkesztő:
Faklen Pál

Főszerkesztő-helyettes:
Varga János

Szerkesztő:
Jakab Ágnes

Munkatárs:
Sziebig Andrea

A mágneslemez melléklet és a Közkincs szerkesztője:
Verebély Pálné

A Lemezkalauz szerkesztője:
Nagy Gábor

A szerkesztőbizottság tagjai:
Barna László, Boros György, Broczkó Péter, Brüll Károly, Farkas Ernő, Herczeg József, Kassay Árpád, Kónya László, Kovács P. Attila, Pintér Gábor, Vargha Dénes, Vékony Tamás, Villányi László, Zoltai Péter

Szerkesztőség, kiadó és hirdetésszervezés:

VIII., Reguly Antal u. 8.
Budapest 1441
Telefon és fax: 133-1839

Felelős kiadó:
Sebestyén Ilona
ügyvezető igazgató

Cédrus Kiadó Kft

Nyomdai előkészítés:
Tipoprint Kft., Budapest
Nyomtatás:
Zalai Nyomda, Zalaegerszeg
Felelős vezető: Galla József

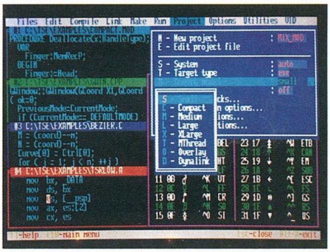
Terjeszti a Magyar Posta. Előfizethető a hírlapkézbesítő postahivataloknál és a Posta Hírlapelőfizetési és Lapellátási Irodájánál (XIII., Lehel u. 10/a, Budapest 1900), vagy átutalással a 215-96162 pénzforgalmi számra.

Példányonkénti ár: 196 Ft
Évi előfizetési díj: 2 352 Ft
PC Turbo Klub-tagoknak: 2 112 Ft
(Tagfelvétel a szerkesztőségben)

Külföldre terjeszti a Kultúra, Pf. 149, Budapest 1389

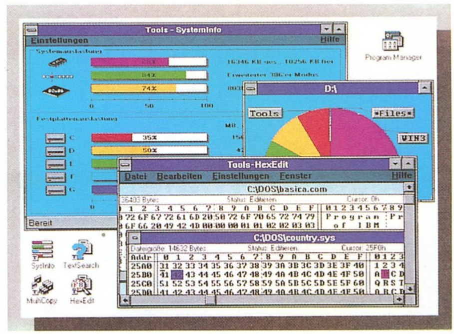
HU ISSN 0865-9788

A HÓNAP TÉMÁJA: DIGITÁLIS BÁBEL



Tolmácsok még kellenek

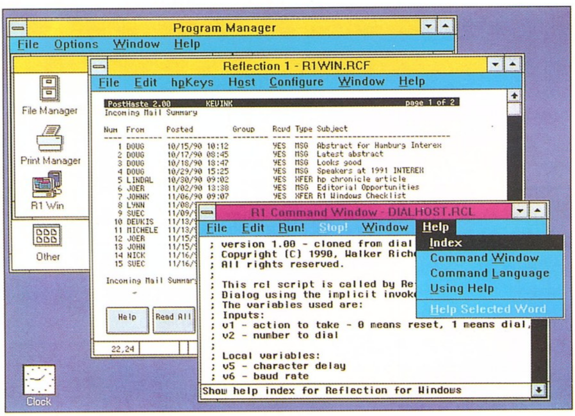
A cél már fél évszázada, a legelső számítógép megalkotásakor is az emberi munka könnyítése volt. Bár a korai gépek tervezésekor a számításigényes feladatok elvégezhetőségét kívánták automatikus útra terelni, azóta a számítógépek már mást is tudnak. Ma nemcsak segítik, hanem sokszor helyettesítik is az embereket. A pionír gépek körül kizárólag szakemberek sürögtek, akik sok mindent „eltűrtek" kedvenceiktől, napjainkban viszont a felhasználók többsége számítástechnikai analfabéta. Ez persze így van rendjén, hiszen nem az embert kell „gépiesíteni", hanem a számítógépet humanizálni.



A két oldal között azonban óriási szakadék húzódik. A számítógépek ugyan ember alkotta eszközök, működésük mégsem hasonlít az emberi gondolkodás és logika — legalább intuitívan — ismert felépítésére. Bár sok kísérlet irányul a gondolkodást modellező gépek megépítésére, napjaink számítógépei csakis az elektronika törvényein alapuló, egyszerű működésre képesek. Ráadásul a gép nem hall, nem érzékel és nem képes tanulni sem, sőt olyan hálátlan is, hogy amit a „szájába rágnak" az egyik nap, azt a másikra már el is felejti.

Míg a számítógép emlékezetkiesésére születhetett hardveres megoldás (a háttértárolók), addig minden egyéb „humanizálás" a számítógépeket működtető programokra hárul. A programok jelentik a számítógép intelligenciáját. A programok egyik fontos tulajdonsága a gép és a felhasználó párbeszédének megkönnyítése, messze van azonban az idő, amikor a gépek kitalálják a gondolatokat. Tolmácsok nélkül a párbeszéd egyelőre még nem megy. Noha kétségtelenül egyszerű a mai grafikus felhasználói felületek kezelése, az egér és az ikon még nem az ember-gép kommunikáció csúcsa — sokak számára ez a „társalgási mód" is idegen. Nem beszélve arról, hogy a legkorszerűbb parancsnyelv-feldolgozó sem képes az utasítás mögött meghúzódó szándékok értelmezésére.

Hogy a számítógépet akármilyen módon is használni tudjuk, szükségünk van valakire, aki a számunkra fontos alkalmazásokat megírja, aki képes a buta hardvert a legbonyolultabb feladatok elvégzésére rábírni. Ez a valaki a programozó. A számítástechnika történetében mégis alig volt olyan idő, amikor nem arról beszéltek, hogy a szoftverek fejlődése messze a hardvereké mögött kullog.



Ami igaz, az igaz: a hardvereszközök teljesítménye óriásit fejlődött. Egy mai programozható zsebszámológép sokkal többet tud, mint akármelyik korai monstrum, miközben energiaellátásához évekig elég egyetlen gombelem.

A számítógépek működését meghatározó elvek azonban 1945 óta változatlanok. Ekkor publikálta Neumann János elképzeléseit a korszerű számítógépekről. Ezen elvek alapján (processzor és a hozzá szorosan csatolt, nagyméretű memória — melyben együtt helyezkedik el a kód és az adat — és a szekvenciális működés) készülnek mind a mai napig a számítógépek. Ezzel szemben a számítástechnika programozással foglalkozó ága folyamatos fejlődésen ment keresztül: a bináris programozástól a 4. generációs eszközökig, az intuitív kódolástól a CASE-ig és a vezérlőpulttól a grafikus felhasználói interfészekig. Ezek közül mi most a programozási nyelvek világába engedünk bepillantást a hónap témája kapcsán. A programozási nyelvek építenek hidat a szárnyaló képzelet világa és az elektronikus működés rideg szabályossága között.

Villányi László

Formulák, listák, algoritmusok a ringben

Klasszikusok — félidőben

Bár a programozástechnika fejlődése során több mint 1000 programozási nyelvet alkottak, azok közül csak tucatnyi terjedt el széles körben. Cikkünkben a ma már klasszikusnak számító nyelvek közül azoknak a rövid történetét foglaljuk össze, amelyek a mikroszámítógépes felhasználók előtt kevésbé ismertek, de mind a mai napig jelentős szerepet játszanak a gyakorlati alkalmazásokban. S lehet, hogy még félidőben sem vagyunk...

FORTRAN

Az első helyen a programozási nyelvek doyenjét, a FORTRAN-t kell megemlítenünk. Bár ez volt gyakorlatilag az első, teljes értékű programozási nyelv, mégis — a megfelelő tervezésnek köszönhetően — töretlen a népszerűsége. A világszerte használatos programok nagy része ezen a nyelven íródott és íródik. (A gáton tehát nem csak két legény van...)

1954-ben az IBM kutatója, John Backus és csoportja merész vállalkozásba kezdett. Olyan automatikus fordítót akartak az IBM 704-es gépre írni, amely matematikai egyenleteket gépi kódra ültet át, és hatékonyságban felveszi a versenyt a kézzel kódolt programokkal.

Ezt a vállalkozást abban az időben meglehetősen nagy szkepticizmussal figyelte a kódolók társadalma, mivel a korábbi kísérletek sorra kudarcot vallottak. Így érthető módon a tervezés és megvalósítás során szem előtt tartott egyetlen cél a generált kód maximális optimalizálása volt. Az a felfogás, hogy a programozási nyelvek fő feladata az emberi gondolatok megfogalmazásának és kifejezésének segítése a számítógép számára, meglehetősen háttérbe szorult.

Az első jelentés még 1954-ben elkészült, és ebben leírták a tervezett egyenletfordító (FORmula TRANslator) rendszerre vonatkozó specifikációikat. Ennek a jelentésnek a nagy része leginkább azzal foglalkozott, hogy magát az ötletet „adja el", továbbá annak fejtegetésével, miszerint a FORTRAN használata teljesen kiküszöbölné a kézi kódolást és a debuggolást!!

A nyelv leírása 1956-ra készült el, de még egy évet kellett várni, míg az IBM szállítani tudta a kész verziót. Érdekes módon a felhasználók nem tolongtak az egyenletfordító rendszerért, de Grace Mitchell könyve, amely bemutatta a nyelvet a nagyközönségnek, megtette a magáét, és egyre többen kezdték alkalmazni az új eszközt.

Mint kiderült, Backusnak és társainak a várakozásai a kódgenerálás tekintetében nem bizonyultak túlzónak. A fordító által generált kód valóban vetekedett a kézi kódolásúval, sőt sok esetben még jobbnak is bizonyult. Ezzel a magasszintű programozási nyelvek végleg bevonultak a programozók eszköztárába. A FORTRAN sikerére jellemző, hogy a 60-as évek elejére mintegy 40 különböző, nagygépes FORTRAN-verzió létezett, amelyek bár megpróbálták megőrizni az eredeti elgondolásokat, mégsem voltak egymással kompatibilisek. (A hordozható kód már abban az időben is csak a programozók álmaiban létezett.)

A felhasználók és gyártók érdekében ezért 1966-ban az ANSI megalkotta az első FORTRAN-szabványt: a FORTRAN 66-ot vagy más néven a FORTRAN IV-et. Az 1966-os szabványt aztán újabbak követték, ahogyan a nyelv újabb és újabb funkciókkal és eszközökkel bővült. (A legfrissebb az 1990-es FORTRAN 90.) A FORTRAN sikere egyebek között annak is köszönhető, hogy a szabványosítás folyamán mindig igyekeztek a nyelvet a legújabb programozástechnikai igényeknek megfelelően módosítani — anélkül, hogy az eredeti célokon változtattak volna. Így azt lehet mondani, hogy bár a FORTRAN a programozási nyelv No 1, mégis szinte az összes, későbbi keletű nyelv hatása felfedezhető az éppen aktuális FORTRAN-szabványban.

A FORTRAN mindig is híres volt a jó minőségű optimalizáló fodítóiról és a numerikus problémák elegáns kifejezésére használható eszközeiről. Az amerikai statisztikai és numerikus algebrai csomagok nagy része FORTRAN szubrutinok formájában áll rendelkezésre, és egyre nagyobb a mikroprocesszoros FORTRAN-fordítók száma, különösen a nagy teljesítményű 32 bites 386-os és 486-os Intel processzorokra. A FORTRAN ugyanolyan jól alkalmazható szuper- és miniszámítógépes környezetben, mint a mikroszámítógépekre, és tisztes kora ellenére úgy tűnik, felveszi a versenyt bármelyik újonccal.

LISP

Bár a LISP a legkorábbi programozási nyelvek egyike, koncepciója mégis gyökeresen eltér kortársaiétól. A nyelvet John McCarthy hozta létre 1956 és 1958 között, és akárcsak a FORTRAN kapcsán, a választott számítógép itt is az IBM 704-es volt. Itt azonban minden hasonlóság megszűnik a két nyelv között. McCarthy a LISP-et a mesterséges intelligencia kutatási nyelvének szánta, és ezért semmiben sem hasonlított a korábban született nyelvekre.

McCarthy úgy gondolta, hogy az emberi gondolkodást modellező rendszerek megvalósítására a matematikából jól ismert függvény fogalom a legalkalmasabb. A függvények kombinálásával létrehozott programok feleslegessé teszik a programozók számára a konkrét gépi reprezentálás apróbb részleteinek ismeretét, így munkájuk hatékonyabbá válik. Mivel az ötvenes években a mesterséges intelligencia alapjának a formális logikát tekintették, ezért McCarthy a LISP-et úgy tervezte, hogy alkalmas legyen logikai következtetések levezetésére. A logikai következtetéseket listákkak és a rajtuk végzett műveletekkel ábrázolta; innen ered a nyelv elnevezése is (LISt Processing).

A nyelv első implementációja 1959-ben lett kész, majd rá egy évre megjelent a nyelv leírása is. Az évek során a

LISP a mesterséges intelligencia terén a kutatások kommunikációs nyelvévé vált: a megjelent publikációkban gyakran használják az algoritmusok és kutatási eredmények ismertetésére. A LISP jól alkalmazható olyan MI-rendszerekben, ahol a rendszer működtetése és kiértékelése szabályokkal leírható, például a szakértői rendszerek vagy a nyelvelemző rendszerek esetén.

Bár a LISP nem eljárásorientált nyelv, mégis sok olyan nyelvi elemet tartalmaz, amelyeket a későbbi eljárásorientált nyelvek átvettek. Ilyen például a függvény mint a program alapeleme, a dinamikus adatstruktúrák, a rekurzió, a feltételes és vezérlőutasítások és a memóriagazdálkodás („garbage collection"). Eredetileg a LISP nem rendelkezett adattípusokkal, kizárólag szimbolikus kifejezések kezelésére volt képes. Bár a természetes számok és a velük való műveletek kifejezhetőek listákon végzet műveletek segítségével is, a túlzott sebességcsökkenés miatt már 1960-ban kibővítették a nyelvet számokkal és aritmetikai műveletekkel.

A hatvanas években a LISP gyorsan elterjedt az egyetemi körökben, de — mint ahogy az már a sikeres programozási nyelvek esetében lenni szokott — mindenki a saját céljainak megfelelően módosította. Így a hetvenes évek végére több tucat egyetemi és ipari LISP fejlődött ki. 1978-ban az utahi egyetem kezdeményezésére megkezdődött a LISP nyelv szabványának a kidolgozása. A megszületett nyelv a Common LISP nevet kapta, és bekerült az Amerikai Védelmi Minisztérium három hivatalos programozási nyelve közé.

A Common LISP egy javított és módosított változata egyébként valószínűleg hamarosan ANSI-szabvánnyá válik. A szabvány tartalmazza a LISP objektumorientált kiterjesztését, a CLOS-t (Common LISP Object System), amelyet már jelenleg is sok helyen használnak.

A Common LISP azonban nem elégítette ki az elsősorban elméleti kutatásokat folytató LISP-felhasználókat. Ők előszeretettel alkalmazzák az MIT-en létrehozott LISP dialektust, a Scheme-et. A Scheme — szemben az elsősorban gyakorlati felhasználásokra készült Common LISP-pel — inkább az eredeti elképzeléseknek megfelelő, tisztán a lambda-kalkuluson alapuló modellt valósítja meg. Emellett a rendszer memóriaigénye kicsi, könnyebben és gyorsabban sajátítható el, mint a Common LISP. A Common LISP rendszerek az ipari felhasználók igényeihez alkalmazkodva robusztus és hatékony fejlesztői eszközök. A LISP nyelvet hagyományosan interpreteres környezetben implementálták, azonban az utóbbi időkben megjelentek a LISP fordítók, de léteznek hardvereszközök is a LISP-rendszerek gyorsítására — speciális processzorok és architektúrák formájában.

Bár a korábbi LISP-implementációk meglehetősen lassúak voltak és hatalmas memóriakapacitást igényeltek, a hardver terén tapasztalható fejlődés hatására (nagyobb tár- és processzorkapacitás, alacsonyabb ár) az eddig leginkább kísérleti rendszerekben használt nyelv újabb alkalmazási területeket hódíthat meg, akár a személyi számítógépek világában is.

ALGOL

1958-ban Zürichben konferenciára gyűltek össze az európai számítástechnikai szakértők és tengerentúli kollégáik. A konferencia tematikája egy általános célú programozási nyelv tervezése volt. Mint általában, a konferencia lényege a nemzetközi szakemberek közötti véleménycsere volt: míg az európaiak elsősorban az elméleti kutatásokban jeleskedtek, addig amerikai kollégáik már több konkrét nyelvi elképzelés megvalósításáról tudtak beszámolni. Egy másik ki nem mondott, de mindenki által tudott cél a FORTRAN esetleges egyeduralmának a megakadályozása volt. Mivel akkoriban a FORTRAN még az IBM tulajdonát képezte, így a nyelv használata egyben az IBM gépek preferálását jelentette volna, ami már akkoriban sem nyerte el mindenki tetszését. A létrehozott nyelvet eredetileg IAL-nek keresztelték el (International Algebraic Language), de végleges verziója az ALGOL (ALGOrithmic Language) néven vonult be a programozási nyelvek történetébe. Az ALGOL 58-at — ahogyan a nyelvet hivatalosan nevezték — az akkoriban létező, szinte valamennyi számítógépre implementálták, és hatására rengeteg új nyelv született.

1960-ban Párizsban ismét összeült a bizottság, és módosították a nyelvet (ALGOL 60), megtartva az 58-as elgondolásokat. Az ALGOL 60 hatása a számítástechnikára és a programozási nyelvek fejlődésére egyedülálló. A sok fontos jellemző közül az egyik a BNF metanyelv választása a nyelv definiálására, amelyet John Backus és Peter Naur tervezett. A BNF (és később az EBNF) jelölésmód azóta a számítástechnika sok területét befolyásolta, mi több: új diszciplínákat hívott létre.

Bár az ALGOL 60-at elsősorban Európában alkalmazták gyakorlati nyelvként, mégis mind az amerikai, mind az európai szakirodalomban szinte kizárólag az ALGOL-t használták és használják mind a mai napig az algoritmusok és általános programozástechnikai elgondolások szemléltetésére (az utóbbi időkben gyakran találkozhatunk a Pascallal és a C-vel is, de ezek is ALGOL-utódok). Magának az ALGOL nyelvnek a befolyása a későbbi programozási nyelvek koncepcióiban egyetlen más nyelvéhez sem hasonlítható. Hogy csak a legfontosabbakat említsük (hozzávetőleges időrendben): COBOL, SIMULA, PL/1, Pascal, C, Smalltalk, Modula-2, Ada. Ezek a nyelvek mind rendelkeznek egymásba ágyazható blokkstruktúrával, összetett utasításokkal, láthatósági szabályokkal, lokális változók deklarálásával és különböző paraméterátadási módszerekkel.

1968-ban az ALGOL-bizottság egy újabb, igen jelentős módosításokat tartalmazó változatot dolgozott ki: az ALGOL 68-at. Az ALGOL 68 óriási, általános célú programozási nyelv — szemben az ALGOL 60 tudományos orientáltságával. Éppen az új verzióban bevezetett, lényeges változtatások miatt a nyelv sohasem lett igazán sikeres. A kritikusok szerint túl nagy és bonyolult az ALGOL 60 egyszerűségéhez képest. Nem meglepő tehát, hogy az ALGOL 68 ugyanarra a sorsra jutott, mint hírhedt társai: a PL/1 és később az Ada is. (Van erre egy jó magyar közmondás: „Sokat akar a szarka, de nem bírja a farka".) Persze az ALGOL 68 nem múlt ki nyomtalanul: itt alkalmazták először a felhasználó által definiált adattípusokat és a mutatótípusokat, amelyek azóta több nyelvben is megtalálták a helyüket.

COBOL

Bár a mikroszámítógépes társadalom számára a „klasszikusok" közül talán a COBOL a legkevésbé ismert programozási nyelv, azonban a nagygépes felhasználóknak még napjainkban is több mint kétharmada részesíti előnyben a többivel szemben. A ragaszkodás persze nem véletlen, hiszen — akárcsak a FORTRAN esetén — a COBOL-ban megírt alkalmazások száma is nagy. Egyedül az Egyesült Államokban 70 milliárd sorra teszik a még ma is használatban lévő COBOL programok méretét.

1959-ben, egy évvel az IAL-bizottság ülése után, egy újabb bizottság alakult meg, ezúttal elsősorban ameri-

kai résztvevőkkel, hogy az üzleti információs és adatbázisrendszerek kezelésére egységes koncepciót dolgozzon ki. A CODASYL (COnference on DAta SYstems Language) tagjai között voltak a kormány, a számítógépgyártók, az ipari felhasználók és az egyetemek képviselői. A cél olyan nyelv kidolgozása volt, amely lehetővé teszi a gazdasági életben felmerülő információfeldolgozási feladatok megoldását kevés számítástechnikai ismerettel rendelkező menedzserek számára is. A tervezett felhasználók köre miatt alapvető fontosságú volt, hogy a nyelv utasításai és a létrehozott programok minél érhetőbbek legyenek a laikus szakemberek számára, ezért a nyelv több száz angol szót tartalmaz, sőt: bizonyos esetekben szinonimák használata is meg van engedve. Ezért a COBOL programok nagyon jól olvashatók, de igen sok gépelést igényelnek.

A CODASYL-ban az Amerikai Védelmi Minisztérium is képviseltette magát, azonban ami még ennél is lényegesebb volt a COBOL jövője szempontjából: egy „ajánlásban" a minisztérium a gyártók tudomására hozta, hogy csak olyan számítógéprendszerek vásárlását és használatát engedélyezi a projektjeiben, amelyek rendelkeznek COBOL-fordítóval. Ez a tény, és az, hogy a COBOL-t eleve géptől független nyelvként tervezték, érthetően nagyot lendített a nyelv elterjedésében. A COBOL, akárcsak a FORTRAN, sok módosításon ment keresztül az évek során. A legjelentősebbek az 1968-as, 1974-es és 1985-ös módosítások. Bár a legutóbbi verziónak még alig volt ideje elterjedni, máris újabb módosításokon gondolkodik a CODASYL, mégpedig az objektumorientált eszközök bevezetésével. Bár a COBOL-t már nagyon sokan eltemették az utóbbi években — elsősorban a 4. generációs eszközök megjelenésekor —, azonban a már élő óriási kódmennyiség és a nyelv folyamatos megújulása miatt az ObjectCOBOL, ha nem is a mikroszámítógépeken, de a nagy rendszerekben még hosszú ideig meghatározó programozási nyelv lesz.

Smalltalk

Az utóbbi időkben már-már hisztérikusnak nevezhető OOP-láz, amely nem utolsósorban az AT&T-nek a C++-hoz fűződő üzleti érdekeiből származik, a PC-s programozók és felhasználók egyre szélesebb táborával ismertet meg egy immáron húszévesnél is korosabb tervezési technikát. A C++ és más, objektumorientált bővítésekkel ellátott programozási nyelvek körüli felhajtásban érdemes néhány mondatot szentelni e nyelvi forradalom egyik elindítójának, Alan Kaynek és a Smalltalknak.

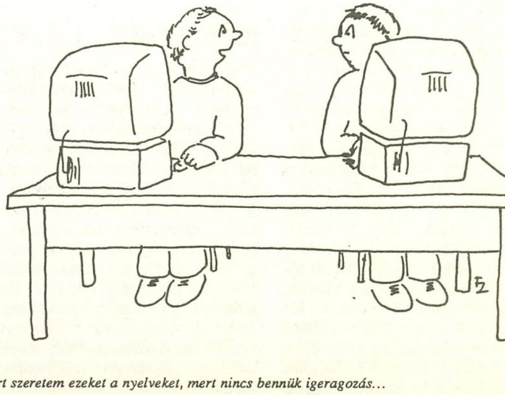
A Smalltalk az a programozási nyelv, amely bevezette és mind a mai napig meghatározza az objektumorientált programozás alapvető fogalmait. Az 1960-as években Alan Kay, az utahi egyetem egyik diákja egy olyan, könyvnyi méretű számítógépről álmodozott, melynek kezeléséhez nem kell főiskolai végzettség, noha olyan szoftverrendszere van, amelyik a felhasználó igényével együtt nő. A gép tervezett szoftvere a FLEX (FLexible EXtensible language) elnevezést kapta, és Kay 1969-ben elkészült diplomamunkájában dolgozta ki a részleteit. Kay az egyetem után a Xerox PARC-hoz került, és szabad kezet kapott elképzelései megvalósításához. Mivel azonban akkor még a noteszgépekhez szükséges hardverkapacitás nem állt rendelkezésre, Kay és munkatársai az Alto munkaállomáson implementálták a rendszer szoftverelemeit Smalltalk néven. A Smalltalk egyesítette magában a FLEX, a SIMULA és a Logo programozási nyelvekben meglévő koncepciókat.

A FLEX az integrált grafikus felhasználói felületet és a rendszer felhasználó általi bővíthetőségét adta, a SIMULA az osztály fogalmát mint eszközt a bővíthetőség megvalósításához, a Logo pedig a könnyen kezelhető grafikus kapacitásokat és a teknőcgrafikát a felhasználói felület kialakításához. Az elkészült Smalltalk rendszer és az Altóval kapcsolatos más projektek egyébként óriási hatással voltak és vannak a személyi számítógépek operációs rendszereivel és felhasználói felületeivel kapcsolatos elgondolásokra. Az olyan fogalmak, mint az ablak, az interaktív menüvezérelt környezet és egér, mind-mind megtalálhatók voltak az Altón. A Smalltalktól függetlenül megvalósítható felhasználói környezettel először az Apple Lisa, majd később a Macintosh gépek felhasználói ismerkedhettek meg először, de újabban a PC-s felhasználók is találkozhatnak vele a GEM vagy a Windows képében.

A Smalltalk 1972 októberében egy 1000 soros BASIC program formájában „látta meg a napvilágot", majd nem sokkal ezután assemblyben írta meg Daniel Ingalls, aki a későbbi implementációkat is gondozta. A nyelvet a PARC berkein belül egészen 1979-ig kétévenként módosították, és újra is írták anélkül, hogy a nyelvet szélesebb körben nyilvánosságra hozták volna. Végül 1979-ben az új verzió előkészítéséhez már külső szakembereket is bevontak, és a korábbi gyakorlattól eltérően a nyelvet már nem speciálisan az Altóra, hanem tetszőleges számítógépre tervezték. A megszületett Smalltalk-80 nyelv azóta is gyakorlatilag változatlan formában a nyelv hivatalos szabványa, és immár mindenki számára elérhető.

Az IBM PC-re először a Smalltalk/V-t implementálták, de újabban a Smalltalk-80 környezet is fut a Windows alatt. Jóllehet a Smalltalk jelenlegi formájában elsősorban oktatási és kísérleti rendszernek tekinthető, nem pedig programozási eszköznek, azért a nyelv befolyása a korszerű programozástechnikára kétségbevonhatatlan, és valószínűsíthető, hogy ezt a befolyását a jövőben csak növelni fogja.

Villányi László



— Azért szeretem ezeket a nyelveket, mert nincs bennük igeragozás...

A 4. generációs programozási eszközökről
Nyelvhasználat — pizzában elbeszélve

A 4GL (4th generation language) címke, az OOP-hez hasonlóan, az utóbbi időkben a legkülönfélébb termékek reklámjában fellelhető — a termék eladhatóságát biztosítandó. A termékek legtöbbje azonban nem sorolható a klasszikus értelemben vett programozási nyelvek közé, hanem inkább a 4. generációs technikával kapcsolatos fogalmakat megvalósító eszköznek tekinthető.

A negyedik generációs eszközök legfontosabb ismérve, hogy működésük központi kérdése nem a hogyan, hanem a mit. Ez növeli a felhasználók munkájának hatékonyságát, mivel egy probléma megoldása során nem kell törődniük a technikai részletekkel, a megoldás menetével, csupán a kívánt eredményt kell meghatározni, minden más a rendszer dolga. A klasszikus nyelveken a programozás során a problémamegoldás minden apró mozzanatát ki kell dolgozni, a programok nagyok, nehezen olvashatóak, és sokszor csak a szerző által értelmezhetőek. A nem eljárásorientált nyelvekben viszont a felhasználónak csak a feladatot kell a rendszerrel közölnie, a részletek kimunkálásáról a szoftver gondoskodik.

„Táplál(koz)ás"

A különbségek szemléltetésére vegyünk most egy példát a gyakorlati programozástechnika területéről. A programok készítése során alapvetően fontos a programozók életben tartása, azaz etetése. A megrendelő szerencséjére azonban a programozók általában képesek ezt a feladatot önállóan elvégezni, csupán a felmerülő költségeket számolják el. Nézzük, hogyan is történik ez a gyakorlatban. A programozó tipikus éjszakai „tápja" a pizza, egy üveg sörrel öblítve — mint köztudott, a programozó éjszakai lény. A klasszikus programozó a pizza előállítását a receptkönyv szerint végzi, lépésről lépésre haladva a tészta, majd a töltelék elkészítésével, utána a sütéssel. Ha programozónk Assembly-alkatú, akkor a sört is sajátkezűleg készíti valamilyen régi bajor recept alapján. Az elkészült pizza és sör minősége nagyban függ az előállítás során elkövetett hibáktól és a programozónak a pizza- és a sörkészítésben való jártasságától. Ha programozónk egy kicsit rutinosabb, és már tisztában van a mások által elkészített könyvtármodulok hasznosságával (C, Modula vagy Pascal hívő), akkor a mélyhűtőben rendelkezésre állnak a maga, a mamája vagy más programozó által elkészített tészta- és szószadagok, valamint a boltban vásárolt sör. A pizzába a tálalásig természetesen még így is kerülhet hiba, de ez most már csak a sütésre és a töltelékre vonatkozik. Az objektumorientált metodikát használó programozó a mélyhűtőben mirelit pizzát tárol. Feladata kizárólag a pizza felmelegítése és utóízesítése, itt alig lehet hibázni. Vagy: a 4. generációs eszközöket alkalmazó programozó felveszi a telefont, a sarki pizzasütőtől megrendeli a kedvenc pizzáját és egy üveg sört. Hát — pizzában elbeszélve ennyi a különbség.

Kevés szóval

A tisztán 4. generációs programozási nyelvek (például PROLOG, Smalltalk) egyetlen „hátránya" a felhasználóval való kommunikációs nehézség (bár a Smalltalk kiváló fejlesztési környezettel bír, ez azonban inkább oktatási és kutatási feladatok megoldására való). Mivel valóban nyelvek, ezért használatukhoz elengedhetetlen a nyelvi szabályok ismerete (szemantika, szintaktika), ami a jártasság megszerzését nehezíti. Ezért a 4. generációs eszközök legtöbbjének van természetes nyelvi interfésze, és ez sokszor kiegészül menüorientált kezeléssel. A felhasználó az óhajait akár ékes angol nyelven is megfogalmazhatja, a rendszer nyelvi processzora képes a lényeg kihámozására. A kommunikáció szabályai így rendkívül egyszerűek, és azokat a nem számítástechnikai szakemberek is könnyen elsajátíthatják. Az így kialakított programok rövidek, jól érthetőek, és karbantartásuk, javításuk is gyors. Persze a jelenlegi rendszerek még távolról sem tökéletesek, ezért az eszközök legtöbbje egy vagy több eljárásorientált nyelvet is tartalmaz.

Mindemellett elgondolkodtató, hogy bár a 4. generációs rendszerek népszerűsége növekszik, az igen nagy szoftverrendszerek elkészítésére 4GL segítségével még nem volt példa. Ellenkezőleg: a szakirodalom nyilvántart olyan konkrét próbálkozásokat, amelyeket végül is 3. generációs nyelvvel kellett megoldani.

A 4. generációs rendszerekben nagy szerepe van az adatbázisnak. Ezért az adatbázis-kezelés a legtöbb rendszer központi része. De az adatbáziskezelő rendszerek korántsem 4. generációs eszközök. Az adatbázissal kapcsolatban a legfontosabb feladat az adatok különféle szempontok szerinti feldolgozása és prezentálása, amit a szakirodalom lekérdezésnek nevez. Az adatbázisok lekérdezésére már a 70-es évek elején megszülettek a nagygépes adatbáziskezelő rendszerek lekérdező nyelvei (például Ingres-Quel, Adabas-Natural, SystemR-SQL), amelyek segítségével programozási ismeretek nélkül is manipulálni lehetett az adatbázisokat. Ezek az eszközök korai formáikban még majdnem olyan nehezen voltak megtanulhatók, mint a hagyományos programozási nyelvek, azonban a menüorientált környezetek és a természetes nyelvi processzorok megjelenésével kezelésük nagymértékben leegyszerűsödött. A természetes nyelvi feldolgozás során a program a felhasználó által megadott mondatokban és utasításokban a lekérdező nyelv kulcsszavait keresi, ezekből szintaktikailag helyes lekérdező nyelvi utasításokat állít össze.

Heterogén források

A 4. generációs rendszerek lényeges jellemzője az integrált fejlesztői környezet. Míg az adatbázis.kezelési funk-

ciókör a nagygépes rendszerek „öröksége", addig az integrált környezet a munkaállomások és személyi számítógépek „hagyatéka". Ez a kétféle eredet egyébként lehetővé teszi a 4. generációs eszközök használatát a legkülönfélébb számítógépeken. Részben ezeknek az eszközöknek köszönhetően a korábban alapvető eltérések a különféle számítógépek kezelésében ma már elenyészőek. (A másik fontos tényező a határok összemosásában a teljesítményviszonyok kiegyenlődése.) Az integrált környezet lehetővé teszi a 4. generációs rendszerek különböző egységeinek konzisztens kezelését anélkül, hogy különböző programokat kellene használni. Az ilyen rendszerek a fentieken kívül a legtöbb esetben a következő bővítéseket tartalmazzák: szövegszerkesztési lehetőségek; programgenerátor, amely programvázlatokból teljes, fordítható alkalmazások generálását végzi; grafikus bővítések; adatbázis-kezelő rendszerhez kapcsolódó jelentésgeneráló rendszer; eszközök képernyő és nyomtatási tervek készítéséhez; döntés-előkészítő eszközök, melyek segítségével a felhasználó gazdasági, statisztikai és matematikai elemzést és modellezést végezhet; parancsnyelvi interfészlehetőség 3. generációs nyelvek és eszközök rendszerbe integrálásához.

A legismertebb 4. generációs integrált rendszerek a Focus, a Ramis II, a Nomad2 és a Natural, amelyek legtöbbje nemcsak nagygépes, de PC-s környezetben is használható.

A 4. generációs eszközök széles körű elterjedését leginkább nagy processzor- és lemezkapacitás-igényük gátolta, ami azonban a mikroprocesszorok teljesítménynövekedésével és a hardver árcsökkenésével mára már nem jelent akadályt. Egy másik lényeges ok, ami összefüggésben van az előzővel, a 4. generációs rendszerekkel kapcsolatos alkalmazási eredmények hiánya. Egyelőre úgy tűnik, hogy sikeresen használják őket a közepes méretű alkalmazások esetén, azonban a nagy rendszerek megvalósítását még inkább 3. generációs nyelvekkel végzik. Ez részben a 4. generációs eszközök specializálódásával, részben pedig az új technikával kapcsolatos ismeretek hiányával magyarázható. Nyilvánvaló, hogy a hardverközeli rendszerek programozásában a klasszikus programozási nyelvek még hosszú ideig egyeduralkodók lesznek, azonban a rendszertervezőknek és programozóknak egyre szélesebb körű ismeretekre lesz szükségük a 4GL, a CASE és a szakértői rendszerek területén.

Villányi László

Két, csak két legény van...

15 évvel ezelőtt — de még tíz éve is —,
ha megkérdeztek volna,
hány programozási nyelv van a világon,
azt feleltem volna, hogy körülbelül másfél ezer.
Ma azt mondom, kettő: a Pascal és a C.

Nem készítettem statisztikát, nem tudom, csinált-e valaki, de az a tapasztalatom, hogy ma az elkészült programok fele C-ben íródott, 40%-a Pascalban, és a maradék tízen osztozik a Basic, Fortran, Cobol, RPG, Modula, LISP, Prolog, Oberon, Miranda stb.

Részekben a finesz, egységben az erő!

Ifjú koromban minden híradástechnikus ismerősöm erősítőt tervezett. Ma inkább kalkulál, és megveszi a legjobbat, amire telik. 15 éve minden alapszoftveres ismerősöm, aki adott magára, megtervezte saját programozási nyelvét, amely bizonyos tekintetben minden korábbi nyelvet fölülmúlt. Azután megszületett az Ada nyelv; ez integrálni akarta mindazt, ami addig a nyelvek területén született. Nem lett sem átütő siker, sem látványos kudarc. Benne van a maradék 10%-ban. Azóta pedig nem látott napvilágot általános érdeklődést kiváltó nyelv. Felmerül a kérdés: mi változott meg akkor és ott, az Ada nyelv tervezése és implementálása során. Én a választ valahogy úgy foglalnám össze, hogy ez volt az az időszak, amikor tisztázódott, hogy mi az az optimális eszközkészlet, amelynek egy nyelvben benne kell lennie. Kiderült, hogy ha ennél többet teszünk a nyelvbe, akkor áttekinthetetlenné, rossz hatékonyságúvá válik. Az úgynevezett omnibusz (=mindenkinek való) nyelvek szerettek volna mindent automatikusan a nyelven belül megoldani: az indexelt fájlkezeléstől a billentyűzet kezeléséig, a valós idejű taszkkezeléstől az osztott adatbázisig, az automatikus szemétgyűjtéssel ellátott dinamikus memóriakiosztástól a diszkek fizikai írásáig/olvasásáig. Nemcsak az igaz, hogy ennyi mindenre ritkán van szükség, hanem sokkal több: egyes feladatok és megoldások eleve ellentmondásban vannak más feladatok más megoldásaival, tehát egyszerre nem alkalmazhatók.

A megoldás a modularitás. Nem a programozási nyelv utasításait kell növelni és egyre körmönfontabbá tenni, hanem a programot világos, jól áttekinthető részekre kell szétvágni. Az egyes részek kapcsolódása legyen tiszta és egyértelmű, és előre elkészített közhasznú modulok széles választéka álljon a felhasználók rendelkezésére. Az input/output procedúrákra, a memóriakezelésre, az adatbázis-kezelésre, a képernyőkezelésre, a billentyűzet kezelésére, a különböző típusú grafikák, az ablak- és menükezelés megoldására, és minden más szükséges területre legyen előre elkészített, jó minőségű modul. Ezek közül azt tesszük hozzá a programhoz, amelyikre szükség van.

Az is kiderült, hogy a nyelv használatában csak egyik összetevő a nyelv, a másik összetevő az IPSE. Az IPSE (integrated program support environment), azaz az integrált programtámogató környezet. Egy jó nyelven könnyebb, gyorsabb programot írni, mint a rosszabbon. De a dolog nem pusztán ezen múlik. A programot meg kell írni, be kell lőni, a verzióit nyilván kell tartani, stb. A belövés során számtalanszor ismétlődik a javítás, fordítás, összekapcsolás, futtatás, megállás, ellenőrzés mint ciklus. Ezt amennyire lehet, automatizálni kell, mert ezzel a munkát gyorsítjuk. Nagy előny, ha a

programozónak nem kell kilépnie a forrásnyelv világából, nem kell regisztereket nézegetni, memóriadumpokat visszafejteni; ha belövés közben azt mondhatja: no, akkor most ennél az utasításnál megállok, és megnézem, hogy amannak a változónak mi az értéke. Hasonlóan nagy kényelem, hogy ehhez nem kell különböző programokból ki- és belépni. Másik könnyítés, ha a program tudja, hogy ha egy bizonyos modult módosítok, mely másakat kell újrafordítani. Ilyen és ehhez hasonló szolgáltatásokat tud egy fejlett IPSE.

Ismerős az IPSE!

Lehet, hogy egyesek most felkiáltanak: de hisz ezt ismerem, ez a Turbo-C vagy ez a Turbo-Pascal. Igen, a Turbo környezet egy IPSE. De nincs egyedül, hasonló környezet a Quick-C, a Logitech Modula fordító, az APSE (Ada Program Support Environment), a CDL2 Laboratórium stb.

Persze nem egészen egyforma ügyesek. Például a nyelvhez kötött editor (amit azért találtak ki, hogy csak a program megváltoztatott részét kelljen újrafordítani) nem volt valami szerencsés ötlet, mert két egymásba ágyazott ciklus felcserélése vagy hasonló „globális" változtatás úgyszólván megoldhatatlan bizonyos rendszerekben. Ilyenkor ki kell lépni, ki kell hozni a program szövegét, egy másfajta editorral át kell szerkeszteni, újra vissza kell vinni a rendszerbe. Persze egy átgondolt programnál erre nagyon ritkán kerül sor.

A C és a Pascal nyelv tehát, amelyek a gyakorlatban leghasználhatóbbnak bizonyultak. A többi nyelv vagy túl kevés, vagy túl sok — sőt, a legtöbb egyes pontokon kevés, másokban pedig túl sok. Kérdés azonban: van-e valami lényeges különbség a két nyelv között? Azt kell mondanom, hogy elvileg (matematikailag) úgyszólván semmi különbség nincs. Ugyanazok az adattípusok, ugyanazok az utasítások, ugyanaz a globális programszerkezet, csak egészen más formában kell leírni. Mi hát mégis a különbség? A különbség a szemléletben van. A C nyelv tipikus angol nyelv: mindent szabad, ami nem tilos. A Pascal „echt" német nyelv: minden tilos, ami nincs megengedve.

A Pascal nyelv szigorúbb, jobban olvasható, mindent explicite ki kell írni, meglehetősen kevés lehetőséget ad a trükközésekre. Ha egy program Pascalban van írva, akkor az egyik gépen így fut, a másikon úgy. A gép fordítóprogramja befolyásolja a program hatékonyságát, de a programozónak ebbe nem-igen van beleszólása. A programozó egy dolgot tud tenni: hatékony algoritmust ír — nyelvtől függetlenül.

A C nyelv voltaképpen feltételezi, hogy a programozó ismeri a gépet, tudja, hogy mit jelent a fordítás — szóval, hogy átlátja a dolgokat. Az igazi C programozó tudja, hogy mikor mit kell kiírni, és ha nem írja ki, mit fog csinálni a fordító. Mikor írhat 4-gyel való szorzás helyett eltolást. Hogyan lehet egy logikai összehasonlítás eredményét közvetlenül aritmetikai műveletekben felhasználni stb. Gondosan ügyel arra, hogy többszörösen egymásba ágyazott ciklusok gyakran futó magjába ne írjon függvényhívást (mert az sok időt vesz igénybe), hanem makrókat használ, ha lehet. Egyáltalán tudja, melyik könyvtári fájlkezelő utasítás függvény, és melyik makró. Sőt, attól sem retten vissza, hogy C programját Assembly nyelvű betétekkel gyorsítsa.

Persze ez nem azt jelenti, hogy muszáj így programozni. Ha szép, áttekinthető (Pascal stílusú), jól olvasható programot akarunk írni, ezt is meg lehet tenni (különösen a C nyelv legújabb változataiban). Hiszen azért a programnyelvek fejlődése nem állt meg. A C nyelv esetében ennek a fejlődésnek a fő jellemzője az, hogy a közérdeklődésre számító programoknál egyre jobban törekszik a jobb olvashatóságra, a típusok jobb ellenőrizhetőségére. Például a régi C feltételezte, hogy minden függvénynek van visszatérő értéke, és ha nem írunk semmit, akkor ez az érték egész típusú. No, de mivel a C nyelvben nincs külön függvény és eljárás, a visszatérő érték nélküli függvényeket, amelyeket pusztán hatásuk miatt használunk, ugyanígy deklaráltuk. Most már lehetőség van arra, hogy az ilyen függvényt „void" típusúnak deklaráljuk, és hibát jelez, ha értékül akarjuk adni.

A Pascal nyelv fejlődése ennél kacskaringósabb volt. Wirth úr annak idején a Pascal nyelvvel együtt egy filozofikus cikket is írt, amelyben élesen állást foglalt a moduláris programozás ellen, arra hivatkozva, hogy a modulok közötti kapcsolatokat nem lehet korrektül ellenőrizni. Ennek megfelelően a Pascal első változatában csak egyetlen programegységet lehetett írni. Persze ez a Pascal csak nagyon kis és nagyon egyszerű programok írására volt alkalmas. A felhasználóknak ez nem nagyon tetszett, és ezért olyan fordítók is születtek, amelyekben mégis lehetett bonyolultabb algoritmusokat is „elereszteni".

Közben az elvi problémát megoldották, és Wirth úr következő nyelvében, a Modulában már a moduláris programozás mellett tört lándzsát. Azóta is folyik a vita az érdekelt körökben, hogy vajon a modularizált Pascal vagy a Modula második változata az igazi nyelv. (Ezzel egy érdekes cikk foglalkozott a Turbo Pascal kapcsán az Alaplap 1991. novemberi számában.) Mivel a két nyelv 98%-ban azonos, a vitát nyilvánvalóan a megszokás és a kellemesebb környezet (IPSE) dönti el.

Tárgyorientált gondolatkörben

Hasonló a helyzet az új csodafegyverrel, az objektumorientált programozással (OOP) is. Az OOP nem talált fel semmi újat, amit Pascalban vagy C-ben ne lehetne megvalósítani. Az OOP egy gondolatkör arról, hogyan kell programot írni. Persze ha ezt a filozófiát programra akarjuk váltani Pascalban és C-ben, akkor egy kicsit többet kell programozni, mint enélkül, viszont a program valószínűleg könnyebben lesz fejleszthető. Ezt a többletmunkát meg lehet spórolni a nyelv továbbfejlesztésével. A kérdés pusztán ez: dobjuk-e ki régi kedvenc nyelveinket, csináljunk-e tiszta lapot, és vezessünk-e be egy olyan nyelvet, amelyben kötelező így programozni, vagy fejlesszük tovább a C-t és a Pascalt kis lépésekben, fokról fokra elkerülve azt a veszélyt, hogy esetleg egyszerre túl nagyot markolunk (és azt a hatékonyság vagy a felhasználó felfogóképessége sínyli meg). A C++, a Turbo C++, a Pascal 5.5 és a Pascal 6.0 láthatólag ezt a másik utat járja.

Természetesen vannak olyan feladatok, amelyekre készen állnak a rendszerek, és a feladat teljesen megoldható. Gyakran ezeknek a rendszereknek is van nyelvük, amelyben az aktuális feladatot le kell írni. Ha azonban olyan feladatunk van, ahol a megoldásnak legalább egy részét a semmiből kell létrehozni, és pláne akkor, ha ezt más meglévő eszközökkel kell összekapcsolni, akkor olyan nyelvet kell választanunk, amely egyrészt a hatékony programozást, másrészt a szükséges interfészt biztosítja. Jelenleg erre a Pascal és a C nyelv esetében a legnagyobb a valószínűség. Ne tévesszen meg bennünket, hogy a feladat első kiírása nagyon egyszerű, és semmi extrát nem kíván; számtalan példa van arra, hogy ha a program első változatában csak fájlokat kellett kezelni, lehet, hogy a következőnek már adatbázist is, avagy ha a program első változata menüvezérelt volt, a másodikba billentyűmakrók is kellenek.

Farkas Ernő

Objektumorientált programozás
A tárgyas „ragozás"

Manapság egyre gyakrabban találkozhatunk az „objektumorientált" kifejezéssel. Van objektumorientált programozás (OOP), tervezés, analízis és adatbázis. De vajon mit is jelent maga az objektumorientáltság — és valóban olyan jelentőségű-e, mint ahogy a megjelenő cikkek és a hirdetések sugallják? Az OOP-metodika jelentőségének megértéséhez kövessük nyomon a programozás történetének vezérfonalát.

A tények sora

Az első számítógépek alacsony sebessége és kis memóriakapacitása arra ösztönözte a programozókat (akiket akkor még kódolóknak hívtak), hogy mindent a gép szemszögéből nézzenek. A programokat elsősorban gépi kódú utasítások sorozatának tekintették. Az adatok csak másodlagos szerepet játszottak, és a tervezés fő eleme a folyamatábra volt.

A klasszikus (3. generációs) programozási nyelvek fejlődése és az alkalmazásgenerátorok, illetve a 4. generációs programozási eszközök megjelenése tükrüzi azt az utat, amelyet a programozás a géporientáltságtól a felhasználóorientáltság felé tett meg. A cél ma már az, hogy a számítógép mint eszköz a felhasználó szempontjai szerint működjék. Mivel azonban a hardverfelépítésben az 50-es évek óta nem volt „rendszerváltás" (a mai gépek még mindig a Neumann-féle tárolt program elven működnek), a változásokat a mind korszerűbbé váló szoftverek jelentették.

A felhasználó által felállított követelmények nagyban függnek attól, hogy konkrétan milyen alkalmazásról van szó. Mást akar a matematikus, mást az üzletember, és megint mást a mérnök. Mások az elvárásai egy számítástechnikusnak, egy titkárnőnek, egy szállodai portásnak. Mást várunk el egy szövegszerkesztőtől, egy táblázatkezelőtől, egy játékprogramtól. Ezért azután a programozási nyelveket és eszközöket az egyes alkalmazói/alkalmazási csoportok szempontjai szerint tervezték, és minél jobban megközelítik az egyik csoport elvárását, annál inkább eltávolodnak a többiétől. Eléggé nyilvánvaló, hogy egy adatbázis-kezelővel senki sem óhajt termelésirányító programot írni. Egyébként ez a dilemma az oka annak, hogy a hagyományos programozási nyelvek mind a mai napig olyan jelentős szerepet játszanak a programozásban, mivel az új eszközök mindegyikét ezen nyelvek valamelyikén írták meg. De hát ha úgyis mindent 3. generációs programozási nyelveken írnak, miért nem használja mindenki ezeket a nyelveket?

Azért, mert ezeknek a nyelveknek — különösen a leghatékonyabbaknak — az elsajátítása sok évi tanulást és gyakorlatot igényel, ami ugye nem várható el egy „mezei" felhasználótól, másrészt a ráfordított idő és energia (hogy ezeknek a nyelveknek a segítségével alkalmazásokat hozzunk létre) olyan nagy, ami egyszerűen nem érné meg. Gondoljuk el például, hogy minden titkárnő maga készítené el saját szövegszerkesztőjét... Beláthatjuk, ez így nem működhet. Mi tehát a megoldás?

Objektumok

Minden alkalmazásban közös az a szándék, hogy bizonyos objektumokat (a valós világ tárgyait vagy fogalmait modellező absztrakciókat) akarunk manipulálni. Programozástechnikai szempontból az egyik eltérés a különböző alkalmazások között az objektumok specifikációja: hiszen ezek például egy nyelvész számára a szavak, a mondatok és a nyelvtani kategóriák lehetnek, míg mondjuk egy üzletember számára a számlák, a rendelések és a tartozások.

A valós világ tárgyainak tulajdonságai vannak, és különböző dolgokat végezhetünk velük. Mondjuk egy lufinak színe, mérete és a térben valamilyen helyzete van, a lufit felfújhatjuk, mozgathatjuk. A számítástechnikai objektumok szintén rendelkeznek tulajdonságokkal (attribútumokkal), és különböző dolgokat lehet velük elvégezni. Ezeknek az elnevezése metódus, operáció, vagy közismertebben művelet.

Az OOP-terminológia nagy része egyébként a Smalltalkból, az egyik legjelentősebb OOP-nyelvből származik. Így például a tartozás objektum olyan metódusokkal kapcsolódik, mint a növekedés, csökkenés és megszűnés. Természetesen nem véletlen, hogy a tartozás objektum attribútumai és metódusai nagyon emlékeztetnek a valódi tartozás tulajdonságaira és viselkedésére — ez az objektumorientált megközelítés egyik erőssége.

Osztályok

A gyakorlatban természetesen nemcsak egyetlen lufival vagy tartozással akarunk dolgozni, hanem lufik és tartozások egész sorával. Az OOP-nyelvek ezért kínálják az objektumtípus vagy más néven osztály fogalmát (ez a hagyományos nyelvek adattípusának kibővített változata). A programozó először tehát egy osztályt definiál a szükséges attribútumokkal és metódusokkal, azután kellő számú példányt hoz belőlük létre. Míg a régi nyelvekben adott típusú változókat deklarálhattunk (például egészszám- vagy karaktertípusúakat), addig az objektumorientált nyelvekben a változók a hagyományos típusokon felül egyes osztályokhoz is tartozhatnak (ezek egyedei lesznek).

Információelrejtés

Az osztályok és objektumok gyakran olyan modulstruktúrák formájában jelennek meg, amelyek az információelrejtés elvén alapulnak. Ez a megoldás ugyan csak félúton van a teljes objektumorientáltság felé, mégis nagyon sok előnnyel jár a hagyományos blokkstruktúrájú nyelvekhez képest. A modulok az objektumorientáltság sok előnyét biztosítják, és megszabadítják néhány hátrányától, ugyanakkor a megvalósításukhoz nem kell objektum-

orientált nyelvet választani, noha e nyelvek közül sok vette át és alkalmazta az objektumokra az információelrejtés elvét. A Smalltalkban például az objektumok attribútumai közvetlenül nem manipulálhatók, csak a definiált metódusok segítségével módosíthatjuk őket. A C++-ban a tervező határozza meg az objektumhoz való hozzáférés módját, ami lehet közvetlen, közvetett vagy ezek tetszőleges kombinációja.

Öröklés

Az objektumorientált nyelvek legfontosabb tulajdonsága az öröklés. A lufi például konkrét fizikai tárgy, így az ilyen tárgyakra jellemző, a köznapi értelemben vett elhelyezkedéssel is mint tulajdonsággal bír. Azaz beszélhetünk a fizikai tárgyak osztályáról, ahol is a fizikai elhelyezkedés az egyik attribútum. Így a lufi osztály a fizikai tárgyak osztályának alosztálya, és mint ilyen, örökli a fizikai elhelyezkedés jellemzőt. Általánosan: egy alosztály örökli az osztály attribútumait és metódusait. A fennálló viszony miatt, illetve az osztályhierarchia szemléltetésére az objektumorientált terminológiában elterjedt az őstípus, előd, utód kifejezések használata. Az osztályok hierarchiája lehetővé teszi a rendszert alkotó objektumok rendszerezését és kézben tartását.

Az öröklés lehetővé teszi a rendszertervezők számára az általánosításokat, és a közös tulajdonságok alapján a rendszer objektumhierarchiájának felépítését. Ez sok esetben egyszerű, de általános célú tervezést eredményez, ami megalapozza a rendszer hosszú idejű használatát erős változtatások nélkül, illetve a karbantartási feladatokat is egyszerűsíti. Nem minden objektumorientált rendszer támogatja az öröklést, vagy ha igen, nem mindig teljes formájában.

Dinamikus összerendelés

Az objektumorientáltság egyik szintén fontos, de még kevésbé általános tulajdonsága a dinamikus vagy futásidejű összerendelés. Ez egy adott művelet és a műveletet megvalósító kód közötti kapcsolatteremtés idejére utal. A tradicionális programozástechnikában a műveletekre és eljárásokra való hivatkozások a program előállítása során, a linkelésnél kerültek a helyükre, és ez a viszony a program futása során nem változott (szerkesztésidejű statikus összerendelés). A korszerű rendszerekben és sok objektumorientált nyelvben lehetőség van a hivatkozások futásidejű kielégítésére, illetve arra, hogy adott helyzetben a rendelkezésre álló eljárások közül egyet válasszunk ki valamilyen szempont vagy mechanizmus segítségével (futásidejű dinamikus összerendelés). Például az objektumok legtöbbjéhez illeszkedik megjelenítési funkció, ez azonban az objektum típusától függően mást jelent. Nyilvánvalóan másként jeleníthetünk meg egy nyulat és egy répát, de eltérés lehet egy szöveges és egy grafikai képernyőt is alkalmazó komplex tervezési rendszerben a betűk megjelenítésében a kétféle monitoron.

Statikus összerendelés esetén a fordítónak, illetve a linkernek a szerkesztéskor tudnia kell, milyen objektumot akarunk megjeleníteni, hogy a megfelelő eljárást beszerkessze a kódba. Dinamikus összerendelés esetén maga az objektum azonosítja az osztályát, és a futásidejű rendszer találja meg az osztályhoz tartozó metódust. Ez utóbbi hatékonyabb és rugalmasabb módszer a statikusnál, és sok OOP-nyelv alkalmazza, sőt: nem kifejezetten objektumorientált rendszerek is. A Smalltalkban minden hivatkozás futásidejű összerendeléssel jár, míg a fordítós nyelvekben (C++, Turbo Pascal) lehetőség van a statikus és dinamikus összerendelés megválasztására a programozás során, ami növeli az előállított kód hatékonyságát. Természetesen a futásidejű összerendeléssel járó méretnövekedést és sebességcsökkenést célszerű figyelembe venni a tervezés folyamán.

OOP — előre van vissza?!

A programozó számára az OOP egyik legjelentősebb előnyét a modulokba való szervezés jelenti, mert lehetővé teszi a rendszerek elemeinek egységes kezelését és az összefüggő részek (adattípus és műveletek) egyszerűbb kapcsolatát. Ezen túl a rendszerkarbantartás és a rendszerbővítés is könnyebb a moduláris architektúrának köszönhetően. A másik lényeges előny a már rendelkezésre álló kód ismételt felhasználhatóságában rejlik. Az öröklés által a megírt modulok/osztályok az új szempontok szerinti tulajdonságokkal bővíthetők, de az eredetiek is módosíthatók anélkül, hogy szükség lenne a kód újraírására vagy módosítására. Így az eredeti struktúrát és elképzeléseket meghagyva, mintegy lecseréljük a rendszer egyes részeit.

Az objektumorientált programozás legnagyobb gyakorlati haszna ez idáig leginkább a szoftvereszközök terén tapasztalható. Az objektumorientált kifejezés mindazonáltal ma még főként vevőcsalogató, mivel az előnyök kihasználásához nagyon sok mindennek meg kell változnia a hagyományos szá-

OOPPÁ!

Mi is tehát az objektumorientáltság, tárgyorientáltság? Egy rendszer, egy módszer, egy technika objektumorientált akkor, ha objektumosztályokon és olyan objektumokon alapul, amelyek mind az attribútumokat, mind a metódusokat magukban foglalják. Az ilyen rendszerek feltehetően, de nem kötelezően kínálják az öröklés és dinamikus összerendelés lehetőségeit.

Az objektumorientált programozás — a strukturált programozáshoz hasonlóan — általános célú programozási metodika, amelyet éppúgy alkalmazhatunk operációs rendszer vagy bérszámfejtő program, mint grafikus program írására. A cél, akárcsak a strukturált programozás esetén, hogy a programozóknak segítséget nyújtson robusztusabb, megbízhatóbb és könnyebben karbantartható programok megírásához. Egy objektumorientált programozási rendszer vagy eszköz segítségével megírt alkalmazás természetesen külsőleg semmiben sem különbözik a hagyományos eszközökkel megírttól — ugyanúgy, ahogy a minőségi jellemzők sem a metodika függvényei.

Mivel egyelőre nem létezik széles körben elfogadott OOP-nyelv (sőt: még a követelmények sincsenek egységes formában kidolgozva), tanácsos, hogy az objektumorientált tervezési módszert szem előtt tartva dolgozzuk ki rendszerünket — anélkül, hogy túlságosan kihasználnánk egy adott implementáció specifikus tulajdonságait. Mindenesetre biztosnak látszik, hogy az objektumorientált programozási nyelvek és programozási környezetek a strukturált programozási metodikát és a nagy rendszerek fejlesztését továbbviszik a XXI. századba.

mítógép-környezetben. Vegyük az operációs rendszerek területét; eddig csak egyetlen, igazán objektumorientált környezet terjedt el relatíve széles körben (a Next gépek Unix-alapú Mach rendszere). A többi operációs rendszernek nincs semmilyen objektumorientált tulajdonsága. A kísérleti rendszereken kívül (Smalltalk, Ceres, Pink) csak bizonytalan lépések tétettek a már meglévőek korszerűsítésére (System 7.0, Windows 3.0, PM).

Gyakran hangzik el olyan kijelentés, hogy az objektumok a szoftverek integrált áramkörei lesznek, pedig ettől ma még nagyon távol vagyunk. Például nincs definiálva az, hogy az eltérő rendszereken — nem is beszélve az eltérő nyelveken — létrehozott objektumokat milyen módon lehet más rendszerekben felhasználni, de legalábbis az egyes egyedeket mágneses adattárolón vagy hálózaton keresztül továbbítani más felhasználók felé. De maradjunk csak a DOS szűkre szabott keretei között, és gondolkodjunk el azon, vajon milyen sikerrel alkalmazhatja egy Borland C++-ban programozó a Turbo Pascal 6.0 segítségével lefordított kódunkat! Sajnos valóság: az objektumok csak egy adott rendszeren belül — mi több: csak egy adott fordítóprogram által létrehozott program működéséig — léteznek, és ebből a keretből egyelőre sem térben, sem időben nem léphetnek ki. Bár a JPI cég TopSpeed nyelveinek új verziója (egy nagyon jó tervezési elgondolásnak köszönhetően) már lehetővé teszi az objektumok használatát C++, Modula ás Pascal környezetben, ez csak a JPI termékeire és az IBM PC-vel kompatibilis gépek környezeteire (DOS-Windows, OS/2-PM) vonatkozik. Egy fecske még nem csinál nyarat.

Egy másik nagy gond az öröklésből és a dinamikus összerendelésből fakad. Mire végre eljutottunk oda, hogy a statikus linkerek legjobbjai (Stony Brook, TopSpeed) már kizárólag csak a program által használt változókat és kódrészeket tartalmazó kódot állítanak elő, máris itt vannak az objektumorientált nyelvek tulajdonságaiból származó problémák. Mint láttuk, egy jól tervezett rendszer- vagy objektumkönyvtár objektumosztályok hierarchiájából áll. Igen ám, de a dinamikus összerendelés nehéz helyzet elé állítja a linkert, mivel a futásidőben fennálló viszonyok elemzése szükséges ahhoz, hogy a végleges kód valóban csak azokat a metódusokat tartalmazza, amelyek kellenek. Amikor a legtöbb linker még a statikus viszonyok elemzését sem képes tisztességesen elvégezni, ez talán túlzásnak tűnhet, ugyanakkor jogos az az igény is, hogy ha az ember egy barackot óhajt elfogyasztani uzsonnára, ahhoz ne kelljen az egész kertet hazavinnie. Objektumorientált szoftvereszközt gyártó cégek közül többen is azzal hirdetik termékeiket, hogy nincsen közös őstípus, minden osztály egyedi, így a futásidejű programméret kisebb, mint egyetlen összefüggő hierarchia esetén. Ez a megoldás viszont meglehetősen eltér az objektumorientált metodika egyik fontos alaptételétől, hogy egy rendszer az objektumok jól megtervezett hierarchiájából épül fel. Így kár erőltetni az objektumorientáltság látszatát (a jobb szlogen érdekében). Ugyanis 3. generációs eszközökkel is lehet kiváló programokat alkotni. A klasszikus Smalltalk nyelvben például minden osztály az Object őstípusból származik, de említhetném pozitív példaként a Borland cég TurboVision könyvtárát is.

A harmadik problémát a klasszikus objektumorientált fejlesztői környezetek speciális tulajdonságaiból származó gondok jelentik. Az ilyen rendszerek (például a Smalltalk-80 és a Smalltalk/V, illetve az Actor) interaktív és interpreteres rendszerek, ami kezelésüket és felhasználóbarát tulajdonságaikat növeli, ugyanakkor mondjuk a Paksi Atomerőmű vezérlését már nem nagyon lehetne rájuk bízni. Az ilyen jellegű igények kielégítésére születtek a 3. generációs nyelveket objektumorientált kiterjesztéssel ellátó megoldások (C++, Objective-C, Object Pascal, az új generációs Modula-2 és Pascal-fordítók, továbbá legújabban az Object Cobol), illetve néhány új nyelv, amelyek közül a legnagyobb figyelem egyelőre a Modula-3, illetve az Oberon felé irányul. (Lásd a 91/12. szám 74. oldalát.)

Alkalmazási területek

Az alkalmazási területek közül talán a felhasználói felületek előállításánál használják leginkább az OOP-t. Elég magától értetődő az összefüggés a képernyőn megjeleníthető grafikus objektumok és a programban található objektumok között. Ez az összefüggés még szorosabbá tehető, ha mondjuk egy egérkattintással a grafikus objektum megfelelő részén egy olyan menüt kapunk, amely az objektumra alkalmazható műveleteket tartalmazza. Az OOP elterjedése a grafikus felhasználói felületek előállításában persze nem a véletlen műve. Mind a grafikus felhasználói felületekkel, mind az objektumorientált rendszerekkel kapcsolatos kutatásban élen járt a Xerox Palo Altó-i kutatási központja (a PARC), és az ott elkészült Smalltalk rendszer egyesítette magában az elért eredményeket.

Természetesen az OOP nem kizárólag grafikus alkalmazások előállítására alkalmas. Mivel a legtöbb rendszer alapvetően információk és adatok tárolásával és megjelenítésével foglalkozik, ezért az objektumok tárolásával (objektumorientált adatbázis) és megjelenítésével foglalkozó integrált környezeteknek nagy szerepük van nem időkritikus alkalmazásokban.

Villányi László

Generációváltás

Az első és második generációs programozási eszközök még alig voltak többek, mint a mindennapi kódolást megkönnyítő segédletek. A harmadik generációs programozási nyelvek már valóban új minőséget jelentettek, nemcsak a kódolás, hanem a rendszertervezés területén is. A harmadik generációs programozási nyelvek száma a kétezret is megközelíti, és noha egyre több a negyedik generációsnak tekinthető programozási nyelv és eszköz, a klasszikus nyelvek ideje még korántsem járt le. Az alkalmazások nagy része még mindig ezeken a nyelveken készül, sőt a negyedik generációs újdonságok zömét is harmadik generációs nyelven írták.

Az aktuális cél azonban — a hardverárak csökkenésének és a nyers számítási teljesítmény hihetetlen növekedésének köszönhetően — nem a gépi kapacitás maximális kihasználása, hanem az emberi erőforrásokkal való optimális gazdálkodás. A programozás aprólékos részleteit maga a számítógép veszi majd át, a programozó feladata így elsősorban a tervezés lesz, amiben ugyancsak számíthat a számítógépre. Azért korántsem kell még temetni. Az egyelőre egyeduralkodó technológiák nem adják meg könnyen magukat, és valószínűleg a jövőben is szükség lesz rájuk.

Szoftvermérnökség

CASE-i vezérlés

A jó programok nem a véletlen szülöttei, és nem is egyszerűen a józan paraszti ész termékei. Az érdemi programokat tervezik. A programtervezési metodika a 60-as évek eleje óta egyre nagyobb hangsúlyt kapott a rendszertervezésben: kezdve a korai intuitív programozástól egészen a napjainkban már automatizált tervezési környezetekig.

Nagy utat járt be a számítástechnika az 50-es évek lyukkártyás, oktális kódolású programjaitól a 80-as évek személyi számítógépes forradalmáig. A számítógépek egyre nagyobb teljesítményűek, a programok pedig egyre terjedelmesebbek és bonyolultabbak. A programtervező feladata ennek a komplex rendszernek a kézben tartása és menedzselése. A szoftvermérnökség magában foglalja a szoftverek analízisét, tervezését, tesztelését, hitelesítését és karbantartását. Ez a számítástechnikai diszciplína körülbelül a 70-es évektől létezik, és kialakulásában nagy szerepe volt a strukturált programozási metodikának. A strukturált programozás nagy lépést jelentett a korai kipróbálás, hibajavítás és spagettikód-technológiákhoz képest, és általánosan elfogadott metódussá vált, amelyet sikerrel lehetett alkalmazni a legkülönfélébb esetekben.

A modern komplex rendszerek esetén azonban a strukturált programozás már nem nyújt elegendő segítséget. Ezért azután a tervezőknek újabb módszerek után kellett nézniük. Ezek közül a legfontosabbak az adatabsztrakció, az információelrejtés, a strukturált rendszeranalízis, az adatbázis-metódusok, az adatbázis-központú 4. generációs eszközök, az objektumorientált programozás és az automatizált dokumentálás. A számítógéppel segített szoftvermérnökség (CASE) mint technika ezeknek a módszereknek nagy részét tartalmazza és automatizálja.

Mindezen módszerek közös jellemzője az, hogy megpróbálják a rendszer komplexitását emberi mértékűre redukálni. Természetesen ez bizonyos megkötést jelent a programozó számára, ugyanakkor viszont gyorsabbá teszi az implementálást. A legújabb módszerek többsége számol azzal is, hogy az elkészült programok élettartama várhatóan hosszú lesz, ezért eleve figyelembe veszik a fejlesztés és karbantartás igényeit is.

A szoftvermérnökség fogalmai közül összefoglaltuk a legfontosabbakat. A közös ezekben a felmerült probléma bonyolultságának redukálása, és így a programozói feladat könnyítése.

A CASE-termékek a programozók által jól ismert eszközöket ötvözik egybe egy integrált környezet segítségével. Ilyen eszközök a fordító, a linker, a debugger és az editor. Ezenfelül a CASE-eszközök a strukturált módszerek, a rendszeranalízis és a szoftvermérnökség automatizált változatait is tartalmazzák. A CASE rendszerek sok esetben ezen túl a specifikáció és a szoftvermérnöki munka alapján a megfelelő kód generálására is képesek a felhasználó által megadott programozási nyelven.

Noha a piacon létező CASE-termékek meglehetősen eltérően értelmezik feladatuk mibenlétét, az mindegyiknek célja, hogy lehetővé tegye a szoftvermérnökök munkájának nagy részét az analízisre és a tervezésre fordítani, nem pedig magára a kódolásra. Ez a jelenlegi tendenciákat figyelembe véve jelentős segítség, hiszen egy teljesen és korrektül definiált probléma megoldására született programtervnek minden esetben jól működő programot kell eredményeznie, tehát a kódolási és tesztelési szakasz jelentősen lerövidíthető.

A CASE-eszközök másik fontos előnye a nagy rendszerek karbantartásában és fejlesztésében nyújtott segítségük. A komplex rendszerek esetén ugyanis a programok folyamatos változtatásokon mennek keresztül a rendszer tökéletesítése során. A CASE-környezet ilyenkor biztosítja azt, hogy a tervek és a

Modularitás

A 60-as évek eleje óta használt alapfogalom. A strukturált programozás teljes egészében a modularitás koncepcióján alapul. Lényege a modulok függetlensége környezetüktől és a modulok egymáshoz illeszthetősége a rendszer építésekor.

Struktúra

A moduláris programok strukturálhatók. A kialakult struktúra függ az alkalmazott módszertől. A programmodulok közötti kapcsolatok alkothatnak gráf- vagy fastruktúrát (hierarchikus szervezés). Például a top-down módszerrel tervezett programok tipikusan hierarchikus struktúrájúak. A hierarchikus struktúra megfelelő a komplexitás csökkentése szempontjából, azonban a hierarchiában nincs mód a megírt kód újbóli felhasználására, mert a szigorú fastruktúra nem teszi lehetővé a közös modulok használatát.

Információelrejtés

A jól definiált modulok egymástól függetlenek. Egymással csak definiált interfészeiken keresztül kommunikálnak. A kliens moduloknak nincs szükségük a kiszolgáló modul implementációs részének ismeretére (lokális vagy belső változók, algoritmusok stb.). Ezt az információt célszerű a kliens elől elrejteni, és ezzel az egyes modulok integritását megőrizni. Így elkerülhető, hogy a rendszert felesleges információkkal árasszuk el.

Absztrakció

Minden absztrakció az információelrejtés elvén alapul. Az absztrakciók arra szolgálnak, hogy a rendszer felépítésére vonatkozó apró részletektől megkíméljék a tervezőt, akinek így csak a nagyobb egységek együttműködését kell vizsgálnia. Természetesen a teljes rendszer megvalósításakor a részleteket is ki kell dolgozni. Az absztrakció elve a bottom-up programozási metodika alapja. Ennek lényege, hogy a rendszert absztrakciók rétegeiből, mint kisebb nagyobb építőkövekből építjük fel. A szoftvermérnökség megkülönbözteti az adat-, a program- és a vezérlési absztrakciót.

működő programok megegyezzenek, és a dokumentációk is a valós állapotokat tükrözzék.

A CASE rendszerek középpontjában az adatkönyvtár áll, ahogyan a 4. generációs eszközöknél az adatbázis, illetőleg a mesterséges intelligencia-rendszereknél a tudásbázis. Az adatkönyvtár tartalmazza a rendszer objektumainak definícióit és az objektumok között fennálló viszonyokat, valamint az objektumok és a programkód közötti kapcsolatokat. Az adatkönyvtár objektumai lehetnek nyomtatási és képernyőtervek, adatbázis-struktúrák, struktúradiagramok és folyamatábrák.

A CASE rendszerekben a programtervezés és a módosítás a specifikáció szintjén marad, így a specifikációban bekövetkezett változásokat a létrehozott kód is tükrözi. Ez lehetőséget ad a CASE-környezeten belül a rendszerrel kapcsolatos elképzelések tesztelésére is. A specifikációbeli változások hatása végigkövethető a rendszermodellen, és kielemezhető anélkül, hogy a valódi rendszeren változtatni kellene.

A CASE-termékek többsége különféle addicionális modulokkal bővíthető, amelyek közül a legfontosabb a programozási csoportok menedzselésére szolgáló bővítés. Ez lehetővé teszi a komplex rendszerrel kapcsolatos feladatok felosztását különböző programozói csoportok vagy egyedi programozók számára, valamint segítséget nyújt a munka különböző fázisainak összehangolására és ellenőrzésére a csoportok között.

A CASE rendszerek bonyolultságát nagyon jól szemlélteti azok ára. Egy alapkiépítésért körülbelül 50 Turbo fordító árát kell kifizetni, és a költségek egy teljes rendszer esetén ennek a tízszeresét is kitehetik. Ezért aztán a CASE-eszközöket csak az igazán kritikus feladatokhoz veszik igénybe, és bár IBM PC-s környezetben is számos implementáció létezik, ezek nagy részét nem személyi számítógépeken futó rendszerek előállítására használják.

A CASE iránt túl nagy a várakozás, emiatt sokan csalódtak is a használatában. Természetesen a szoftverfejlesztés mindig is nehéz és bonyolult feladat volt, a CASE rendszerek segítségével azonban gyorsabban állíthatjuk elő a szükséges specifikációkat és terveket, továbbá támogatást kaphatunk a kódíráshoz is — hiszen a CASE betűszóban a hangsúly az „A"-n, azaz a támogatáson van. Mindazonáltal a CASE csak az egyik segédeszköz a komplex szoftverrendszerek fejlesztéséhez.

Villányi László

Mesterséges intelligencia

Agyszerű és nagyszerű

A számítógépek megjelenésével szinte egyidős a törekvés: hogyan lehet velük az emberi viselkedést, annak intelligensnek nevezhető jegyeit modellezni, mesterségesen előállítani.

Az ún. mesterséges intelligencia egy lehetséges meghatározása: „A számítástudomány azon részterülete, amely »intelligens« számítógépprogramok kifejlesztésének kérdéseivel foglalkozik. Ezek olyan programok, amelyek problémákat oldanak meg, általában tanulnak a korábbi tapasztalatukból, megértenek természetes nyelvű közléseket, képeket értelmeznek: olyan viselkedést mutatnak, amit az emberek esetében intelligensnek lehetne nevezni." (Idézet a Szakértő Rendszerek '88 című, a Számalk által kiadott tanulmánykötet kislexikonából.)

A következőkben — az effféle kérdésekkel foglalkozó, tervezett TUDÁSTECHNOLÓGIA rovatunkhoz az olvasók várakozását, kíváncsiságát felkeltendő — röviden vázoljuk a mesterséges intelligencia kialakulását, jellemző részterületeit, a fejlődés fő irányait.

A történet kezdetei

Lehetne onnan is indítani, hogy az emberi intelligenciával már az ókorban is foglalkoztak: az emberek gondolkodásának rejtelmeit vizsgáló görögök már közel 3000 évvel ezelőtt létrehozták a logika, a szintaxis és az ismeretelmélet olyan rendszereit, amelyek mindmáig meghatározóak. Ezek — pontosabban az ezekből időközben kifejlődött formális logika, a számítástudomány korai eredményei, valamint a kognitív pszichológia (a megismerés lélektana) — képezik alapját annak az új tudományterületnek, amelyet mesterséges intelligencia néven neveznek már 35 éve.

A (programozható) számítógépek megjelenése a második világháború idejére esik; a numerikus számításokra való alkalmasságuk már ekkor bebizonyosodott. Nem véletlen, hogy a háború befejeztével, az ún. hidegháborús korszak éveiben a gépi fordítórendszerek kidolgozása volt a figyelem középpontjában (amely megoldotta volna a fennmaradt német háborús dokumentumok, majd később az egyre gyarapodó tudományos és technikai dokumentumok fordítását egyik nyelvről a másikra). A kezdeti elvárásoktól azonban az elért eredmények messze elmaradtak.

Bizonyára Alan Turing, a híres angol matematikus „Computing machinery and intelligence" (1950) műve inspirálta a mesterséges intelligencia atyját, John McCarthyt, aki az információfeldolgozó nyelvek első, 1956-os konferenciájára készülve használta először a „mesterséges intelligencia" (MI, angolul: artificial intelligence, AI) megnevezést.

Azok a legendás 60-as évek...

A 60-as évek elején megjelent a LISP, az MI első alapnyelve; a másik pedig a 70-es évek elején megalkotott Prolog. Mindkét nyelv a matematika formális rendszereire (az ún. lambda kalkulusra, illetőleg a matematikai logikára) épül. A kívánt keresési algoritmusokat (lásd alább) nem algoritmikusan, hanem a formális levezetéseket mintázó szimbólum-manipulációk alkalmazásával lehet bennük beprogramozni — emiatt az eddigi MI-programokat szimbolikus programoknak is nevezik.

Az MI első sikerei (a 60-as években) a matematikai logika tételeinek bizonyításában hasznosultak, és a játékelmélet, főleg a sakkjáték területén is megmutatkoztak. Ezek jól körülhatárolható, véges számú szabállyal jellemezhető tárgyterületek. Az ez irányú kutató/fejlesztő munka során mindmáig értékes eredmények születtek a (megoldás)keresési stratégiák, a hatékony keresési algoritmusok területén.

Csakhamar kiderült azonban, hogy nehéz és összetett feladatok számítógépes megoldásához bármilyen bonyolult keresési stratégiát is alkalmaznak, mégsem tudják megközelíteni a jól képzett szakemberek (például sakkmesterek) intuitív feladatmegoldó képességét. Bebizonyosodott, hogy a bonyolultsággal könnyebben lehet megbirkózni, ha magát a feladatot és a megoldásának lépéseit leíró ismeretanyagot ábrázolják megfelelően („jól strukturált" módon) a számítógépben. Ekkor ugyanis annak kezelése, a feladatmegoldási szituációtól függő mozgósítása egyszerű mechanizmusokkal jobban megfogható.

Ismeretbázis plusz következtetés

Az ilyen, ún. ismeretalapú vagy tudásalapú („knowledge based") rendszereknél megvalósuló új programszervezési elv: a tárgyterület szakértőjének ismereteit tartalmazó ismeretbázis és az ezt (szituációfüggően) működtető következtető mechanizmus jól elhatárolódik. E rendszerek többsége a nem számítástechnikusi beállítottságú emberek számára is „barátságos" felhasználói felületet biztosít, mégpedig oly módon, hogy a rendszer működését bárki képes legyen követni: a felhasználó kérésére a rendszer az elvégzett következtetési lépésekhez természetes nyelvű magyarázatot, indoklást is adhat. Az utóbb felsorolt rendszerkomponensek — megfelelő ismeretbázis-feltöltő és -karbantartó szolgáltatásokkal kibővítve — keretrendszerként („üres kagylóhéjként", shellként) jól hasznosíthatók.

A 80-as évek elején megjelentek az első ilyen, ismeretalapú technológiával megépített ún. szakértő rendszerek, majd shellek. Természetesen az MI egyéb részterületei is jelen voltak ekkor már a piacon: az MI-hardverek, az MI-nyelvek, a természetes nyelv feldolgozása, a hangfelismerés/hanggenerálás, a gépi látás, majd (a 80-as évek vége felé) a neurális hálózatok.

A 6. generáció

Az MI jelenleg eredményesen művelt fontosabb részterületei (betűrendben: automatikus programozás, automatikus tételbizonyítás, gépi látás, gépi tanulás, neurális hálózatok, párhuzamos feldolgozás, robotika, szakértő rendszerek, tervezésautomatizálás, valamint természetesnyelv-feldolgozás. (Az Alaplap következő számaiban tervezzük ezen szakterületek bemutatását.) Az MI fenti részterületei közül a gépi tanulás és az automatikus programozás vezet át közvetlenül a jelenlegi (4. és) 5. generációs számítógépes rendszerekből a 6. generációsakba. A továbbiakban erről lesz szó.

Az első számítógépek megépítése óta több generációváltás történt: legyen minél nagyobb tárkapacitású, majd legyen minél gyorsabb, minél kisebb, minél olcsóbb a hardver (utóbbiak a ma közkedvelt PC-k). A Neumann-elvű (hagyományos) gépek ezen négy generációja után a 80-as évek elején a japánok bejelentették az 5. generációs számítógéprendszer létrehozására irányuló munkatervüket. Ez logikai (Prolog-) alapú, nem Neumann-elvű, nagy párhuzamosságú hardver előállítását, valamint azon intelligens információrendszerek kifejlesztését és széles körű alkalmazásbavételét tűzte ki célul. (A fejlesztések filozófiai háttere: akié a vezető szerep most az információfeldolgozás területén, azé lesz a politikai a 21. században.)

Az 5. generációs rendszerek átütő sikere elmaradt; a japánok már 1985-ben kezdtek publikálni 6. generációt megcélzó projektről, amelynek kulcsszava: „brain-like computers" vagy neurális hálózatok. Elméleti alapok: logika, nyelvészet, pszichológia, fiziológia, számítástudomány és mérnöki tudományok. A neurális hálózatok olyan nagy párhuzamosságú dinamikus rendszerek, amelyekkel az eddigieknél jobban lehet közelíteni az (emberi) intelligens információfeldolgozást. E gépek képesek környezethez alkalmazkodó, ún. adaptív működésre is; „programozásuk" történhet tanulással és begyakoroltatással is.

A 6. generációs számítógépes rendszerek természetesen magukba integrálják mindazokat az eredményeket, mindazokat a „hagyományos" és MI-technikákat, amelyek az eddigiek során hasznosnak bizonyultak. Ezen új technológia fogadásához, a fejlesztésekbe való bekapcsolódáshoz tehát mindenképpen szükséges az „eddigiek" ismerete. **Sántáné Tóth Edit**

Egy cikksorozat elé

Naponta mindannyian átéljük, milyen gyorsan nő az az információmennyiség, ami munkánk sikeréhez, mindennapos tájékozottságunkhoz szükséges. A 21. század, az „információs társadalom" küszöbén állunk (sokak szerint máris ott vagyunk). Az igényeknek megfelelő, „emberi módon segítő/tanácsadó", integrált intelligens információs rendszerek segítségével tudunk csak igazán eleget tenni ennek a kihívásnak. Mit várhatunk el ma egy „intelligens" számítógépes rendszertől, milyen számítógépes támogatást fog kapni a holnap embere, és mindezeket milyen harver/szoftver technológia biztosítja? Ezeket a kérdéseket világítjuk meg az Alaplap ezutáni számaiban.

Az MI iránt érdeklődőknek rendszeres fórumot ad többek között az NJSZT Mesterséges Intelligencia, valamint Alakfelismerés szakosztályain kívül több más szervezet is. A mesterséges intelligenciának, különösen a szakértő rendszereknek a hazai szakirodalma is igen bő. Az érdeklődőknek ajánljuk — akár a sorozat előtti tájékozódáshoz, akár az egyes tárgykörök bővebb tanulmányozásához — a hazai konferenciák kiadványait, a CWI (IDG) lapjait, valamint a következő anyagokat:

— Információ-Elektronika 83/2-6, 84/1, 84/3, 84/5, 84/6 és 85/1 számai. (Az 5. generációs projekt anyagainak tematikus feldolgozása.)

— Y. Shiray, J-I. Tsuii: Mesterséges intelligencia. Novotrade, 1987. (Az angol eredetit 1982-ben jelentette meg az Iwanami Shoten Publ.)

— G. L. Simons: Szakértői rendszerek és mikrók. Műszaki Kiadó, Budapest, 1987. (Az angol eredetit 1985-ben kiadta a The National Computing Centre Ltd.)

— Tudomány — a Scientific American magyar kiadása. Számítógép-szoftver különszám, 1986.

— Információ-Elektronika 85/3, 85/4, 85/6, 86/1, 86/2, 86/5, 86/6, 87/1-2, 87/3, 87/4-5, 87/6, 88/1-2, 88/3-4, 89/1-2, 90/1-2 („Szakértő rendszerek Magyarországon '90" tanulmánykötet) és 90/3-4 számai.

— Mikroszámítógép Magazin 87/5 — 88/2 számai.

— Mérés és Automatika 87/12, 88/9, 88/11, 88/12, 89/3, 89/6, 90/3, 90/4 és 90/5 számai.

— Gábor András (szerk.): Szakértő rendszerek '88 — Ismeretalapú információfeldolgozás Magyarországon. Számalk Kiadó, 1988.

— Mérő László: Észjárások. A racionális gondolkodás korlátai és a mesterséges intelligencia. Akadémiai Kiadó, Optimum Kiadó, 1989.

— Fekete István, Nagy Sára, Gregorits Ferenc: Bevezetés a mesterséges intelligenciába. LSI ATSZ, 1990.

A „pixelpreparátor"

Avant Vektor(izáló)

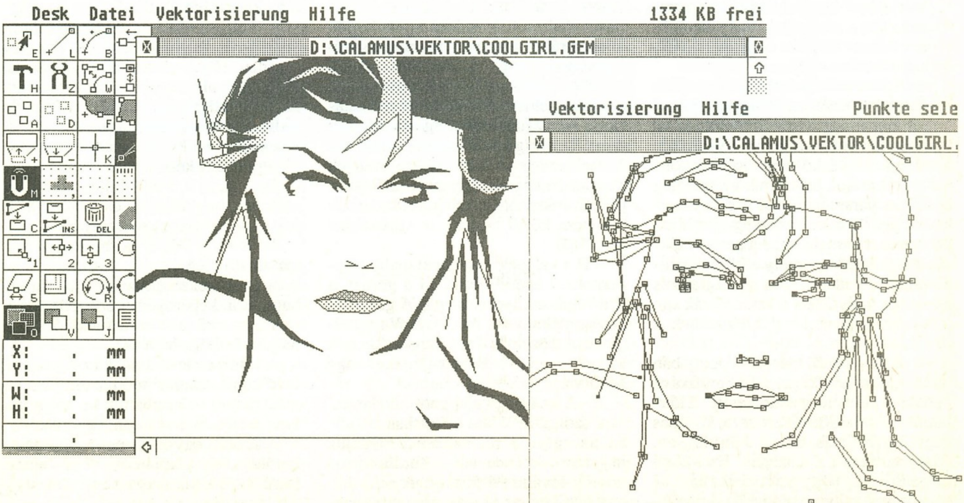
A modern DTP rendszerek egyik nélkülözhetetlen utility programja az, amely a pixelgrafikus képet vektorgrafikussá alakítja át. Ez azért fontos, mert a pixelgrafikus kép nagyításakor a képpontok „meghíznak", ezért csökken a kép felbontása, vektorgrafika esetén viszont méretváltozáskor a program ugyanolyan felbontásban mindig „újrarajzolja" a képet.

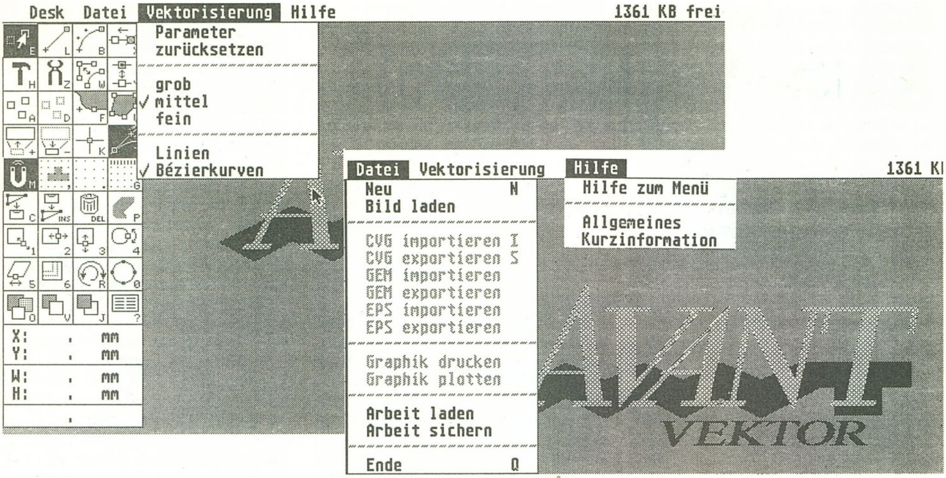
Felvetődik a kérdés: ha ennyivel jobb a vektorgrafika, akkor miért nem eleve ebben a formátumban készítjük el az ábrát? Ennek több oka is van. Az egyik, hogy sok képet könnyebb megrajzolni pontonként, mint előre meghatározott, rögzített alakzatokból, vonalakból. A másik, hogy a szkennerek szoftverjei pixelgrafikát használnak, miután a képet soronként, azon belül képpontonként tapogatják le. Tehát úgy érdemes dolgozni, hogy a szkennerrel beolvasott pixelgrafikus képet retusáljuk (például a Cranach rajzprogrammal), azután pedig egy segédprogrammal átalakítjuk vektorgrafikussá, és véglegessé finomítjuk. Most egy ilyen átalakító segédprogramot ismertetünk, az Avant Vektort, amely már Magyarországon is megvásárolható.

A program Atari ST számítógépre készült. Funkcióival IMG formátumú pixelgrafikus képet tud átalakítani CVG, GEM vagy EPS formátumra. A CVG a Calamus VektorGrafika, a GEM pedig a GEM felhasználói felület standard formátuma, míg az EPS az MS-DOS kompatibilis gépek és a MacIntosh által is használt postscript formátumot jelenti.

A program kétfajta, egymást kiegészítő menürendszert használ: a legördülő (pull down) redőnymenüt (1. ábra), és egy ikonos menüt, amely a képernyő bal oldalán részén helyezkedik el.

A német változatban a Datei, az angolban pedig a File funkció lenyitásával jutunk el a betöltő és az elmentő utasításokhoz. A Neu utasítással új vektorgrafikus képet rajzolhatunk, míg a Bild laden funkció egy pixelgrafikus háttérkép behozatalát teszi lehetővé. Ez lehet éppen egy szkennerrel beolvasott kép is, vagy programmal betölthető IMG, TIF (azaz TIFF, a MacIntoshon és az MS-DOS gépeken használt formátumok egyike), IFF (Amiga képformátum), BLD (a megapaint rajzolóprogram által létrehozott kép), PIC (Atari képernyőformátum), PAC (a Stad rajzprogram képfájlja) és a Degas programmal létrehozható PI3 és PC3 formátum. A későbbiekben ismertetett vektorizáló funkció az így betöltött képből készít vektorábrát.





A CVG, a GEM és az EPS importáló-exportáló funkciókkal az előbb ismertetett vektorképeket tölthetjük be és menthetjük el. A Grafik drucken paranccsal nyomtathatjuk ki az ábrát. A program az Atari lézernyomtatókat és a HP Laserjettel kompatibilis nyomtatókat kezeli, 300 dpi felbontással. A 24 tűs nyomtatók közül a NEC P6-tal kompatibilis, a 9 tűs nyomtatók közül pedig az Epson FX kompatibilisekre készítették fel a programot.

A Grafik plotten utasítás után választhatunk, hogy plotterünk vagy kivágóplotterünk a HP-GL vagy a GP-GL formátumot használja-e.

Az Arbeit sichern funkcióval az előre beállított paramétereket menthetjük el lemezünkre vagy a winchesterre, ez az utasítás létrehozza az AVANTVEK.INF nevű fájlt. Ha legközelebb ismét betöltjük a programot, akkor már az előzőleg beállított paraméterekkel fog bejelentkezni, így nem lesz szükség újra kiválasztani a nyomtató vagy a plotter típusát; megadni, hogy mely alkönyvtárakban helyezkednek el a pixelgrafikus képek, és melyekben a vektorábrák stb.

Az Ende funkcióval kiléphetünk a programból.

A redőnymenü második pontjában találhatjuk a pixelképet vektorgrafikává átalakító funkció paramétereit. Előírhatjuk, hogy a közelítés durva, közepes vagy finom (grob, mittel, fein) legyen. Alapértelmezés a közepes. Ezenkívül megadhatjuk, hogy vektorizáláskor az algoritmus milyen módszert használjon: egyenessel vagy Bézier-görbével dolgozzon-e. Az első esetben a program az ábra körvonalához (állandóan megtörő) egyenes vonalakkal közelít, míg a másik esetben görbékkel. A redőnymenüben a segédinformációk kiírási módját is beállíthatjuk.

Az 2. ábra bal oldali részén jól látható ikonmenüben kínált funkciókat a bal felső saroktól jobbra és lefelé haladva ismertetjük. Minden négyzetben van az ábrán kívül egy betű is, amely jelzi, hogy egér használata nélkül melyik billentyűvel érhető el a funkció.

Az első az E, amellyel kijelölhetjük a vektorábra alappontjait, majd kiválaszthatjuk, hogy a meglévő pontok közül melyekre vonatkozik a következő utasítás (például elforgatás, elmozgatás stb.). Az L funkcióval egyeneseket, a B-vel Bézier-görbéket rajzolhatunk. Az X-szel a megrajzolt ábrából törölhetjük a szomszédos, előző pontot (a Bézier-görbék irányítottak, tehát két szomszédos pont közül csupán az egyik lehet az előző).

A H a kalapács, amellyel újabb alappontokkal egészíthetjük ki a görbét, a Z a fogó, amllyel a meglévő görbéből alappontokat vehetünk ki. A W segítségével a közelítés fajtáját cserélhetjük ki: az egyenest Bézier-görbére vagy fordítva.

Az A az összes alappont kiválasztására szolgál. Például akkor használjuk, ha az egész ábrát máshova kívánjuk helyezni. A D ennek a fordítottja, a pontok kiválasztását törli. Az F a félautomatikus, az U a teljesen automatikus vektorizálást kapcsolja be.

A + a nagyítási, a – a kicsinyítési funkció. A K-val szálkereszt kapcsolható be, a T-vel pedig rajzoláskor a program megjeleníti a görbe érintőjét is.

A következő, azaz az 5. sorban lévő funkciókkal — a rajzolást segítendő — raszterrácsot és vonalzót rajzoltathatunk ki a képernyőre. A mágnes bekapcsolása megakadályozza, hogy a raszterrácson kívülre vagy a rácspontok közé rajzoljunk alappontot. A raszterrács finomsága állítható.

A C billentyűvel az alakzat kiválasztott részét (akár az egészet is) egy segédtárolóba, más szóval egy pufferba tölthetjük, ahonnan később kivehetjük az INS billentyűvel. A DEL funkció törli az E-vel kiválasztott pontokat. A P segítségével ki- és bekapcsolhatjuk a pixelgrafikus képet.

Az 1, 2, 3, 4, 5 és 6 billentyűkkel az alakzatot nagyíthatjuk és kicsinyíthetjük a nyíllal jelzett irányokba.

Az R-rel a Bézier-görbe iránya változtatható meg.

A 0-val sokszöget vagy kört rajzolhatunk a képernyőre. Sokszöget kapunk, ha eközben az L funkciót használjuk, és kört, ha a B-t.

A 9. sorban lévő ikonok az egymáson lévő képek takarási módját állítják. Segítségükkel választhatjuk ki, hogy milyen sorrendben takarják egymást.

A 2. ábra egy szkenner segítségével betöltött kép alapján készített vektorgrafikát és annak vektorizált „csontvázát" mutatja.

Kovács P. Attila

640 kilobájton innen és túl

Mi fér a memóriába — és hova?

Jelen sorok szerzője több mint kétévnyi PC-kompatibilis szoftverkiskereskedelmi tapasztalatai alapján bátran kijelenti, hogy a PC-k memóriakezelésével kapcsolatos ismeretek még a gyakorlott számítástechnikusok körében is meglehetősen hiányosak és homályosak. Több olyan tévhit kering, amelyet egyedileg — egy szoftverbolt pultja mellől vagy telefonon keresztül — szinte lehetetlen eloszlatni. Jó lenne tehát lefújni a misztifikált ködöt a különböző memóriabővítésekről és az ezeket optimálisan kihasználó memóriamenedzser szoftverekről, s minél több embert megismertetni az alapvető fogalmakkal és megoldásokkal. A következő hónaptól kezdve ezért részletesen is bemutatunk néhányat az itt csak megemlített programokból.

Hadd keljek mindenekelőtt a bevezetőben említett szakemberek védelmére. Honnan is tudhatnák, hogyan épül fel hardver- és szoftverszempontból egy IBM PC kompatibilis számítógép? Ilyesmivel alaposan és részletesen foglalkozó magyar nyelvű szakkönyv tudomásom szerint nincs, s a legtöbb ilyen témájú mű csak elnagyolva taglalja a memóriabővítésekkel kapcsolatos információkat, rögtön belevágva az ezeket kezelő eszközmeghajtók (a CONFIG.SYS-ben aktivizálandó device driverek) és segédprogramok parancssori paramétereinek végeláthatatlan felsorolásába.

Memor(al)izáljunk egy kicsit!

Az embernek gyakran az az érzése, hogy a szakkönyveket nem olyanok írják és lektorálják, akik pontosan tudják és értik is, amit írnak, avagy csak gyorsan átsiklanak az angolszász nyelvterület szoftverkereskedőinek szavajárásával piszkos munkának (dirty works) nevezett nem túl lényeges részleteken. Valószínűleg azért tehetik ezt meg, mert bíznak a világ szoftverkommunájának autodidakta ezermestereiben, akik egy ilyen szoftvert egyszer jól beállítanak, s attól kezdve egy fél kontinens a jól bevált recept szerint jár el. Akkor meg minek oldalakon keresztül magyarázni azt, amit úgyis átlapoznak az avatott szakértők is?!

A szoftverkézikönyvek esetében is hasonló a helyzet, maximum a hátsó szószedetben kaphat az ember néhány soros se füle, se farka, nesze semmi, fogd meg jól definíciót. A rendszerprogramozók vakon hisznek a hardver- és szoftverinterruptok mindenhatóságában, és igyekszenek a leírásoknak megfelelően paraméterezve meghívni egy memóriakezelő rutint, bár már ezeket az alacsonyszintű funkciókat is szinte teljesen elfedik a programozók elől a magas szintű programozási nyelvek (C, Pascal, Basic, Modula) fejlesztői környezetei.

Aki keres, az találgat!

Hogy mégis honnan szedegethetők össze azok az ismeretmorzsák, amelyeket most igyekszem az Alaplap olvasóival megosztani? Nos, ha az ember kellően kíváncsi és kitartó, sok érdekes dolgot kihámozhat például az EMS 4.0 szoftverszabványból (a SolarSoft #436 számú lemezén találtam). Egyes külföldi szaklapokból és programlistákból is hasznos összefüggésekre, szabályokra és tényekre bukkanhatunk. Szemléletes ábrákat találhatunk a Quarterdeck cég programleírásaiban is (DESQview, QEMM 386 és QRAM). Szerencsére a számítástechnika empirikus tudomány, azaz a gyakorlat mindenekelőtt, így próba szerencse alapon is sok hasznosítható eredményt lehet elérni, néha persze sok sikertelen kísérletezés, géplefagyás és elfecséreltnek tűnő órák, napok árán.

Emellett kellő kritikával és természetes kétkedéssel megfogadhatjuk a szoftverőrültek megtermékenyítő tanácsait is.

A címzett ismeretlen?

Az Intel processzorok legsúlyosabb öröksége — kezdve az i8088-tól a 8086-on, 80286-on, 80386-on át a 80486-ig — a hardver memóriakezelésében rejlik. Jelesül abban, hogy a rendelkezésre álló memóriát csak 64 kilobájtos lapokon keresztül (16 bit felhasználásával) képesek címezni. Amennyiben a programgyártók meg kívánják őrizni a lefelé irányuló szoftverkompatibilitást is (azaz egy 386-os vagy 486-os processzorra írt program fusson 286-os gépen is), a szoftver architektúrájának (gépi kódban írt BIOS), követnie kell a kezdetek kezdetén kiötölt címzési konvenciókat.

További megkötés volt, hogy egy 286-os előtti processzor csak 20 biten, azaz 1 megabájtig volt képes memóriát (RAM-ot) megszólítani. Mivel a perifériák és az azokat kezelő beégetett programok (BIOS ROM) számára ugyanekkor a 640 kB feletti (A0000 hexa címen kezdődő) területet jelölték ki, így egy közönséges PC-szoftver a DOS betöltése után kezdetben a 640 kB-ból fennmaradó 550-600 kB-ban kellett, hogy megférjen. A 640 kB és az 1 MB között foglal helyet a videomemória. Színes videokártyák esetén a B0000 és a B8000 közötti 32 kB szabadon marad, míg ha a grafikus üzemmódokról is lemond a felhasználó, speciális programokkal az A0000 és a B0000 közötti RAM is felszabadul. A C0000-tól a harddiszket kezelő 32 kB-nyi, míg F0000-tól a 64 kB-nyi BIOS ROM található. A 640 kB és az 1024 kB, azaz 1 MB közötti területet UMB-nek is hívják (upper memory block).

RAM-ban bővelkedve

Egy jelentős adatmennyiséget kezelő program már a 640 kB-os terület határait feszegeti (történetileg ez legelőször a Lotus 1-2-3 táblázatkezelővel esett meg. Magát a szoftverszabványt LIM EMS-ként is emlegetik, amelyben a LIM rövidítést három nagy cég, a Lotus, az Intel és a Microsoft kezdőbetűi adják, miután ők hozták létre ezt az új hardver- és szoftvermegoldást, melynek révén az erre felkészített — de csakis a gyártó által erre felkészített! — programok adataik számára pótlólagos memóriaterületet is fel tudnak használni.

Voltak valaha az egyszerű, 256 kB-os, harddiszk nélküli PC-k és a merevlemezzel is ellátott XT-k. EMS (expanded memory system) bővítőkártyát kellett vásárolni, s az ezen található RAM-ok egy 64 kB-os memóriaablakon keresztül váltak elérhetővé. Fizikailag ezt a memóriaablakot 640 kB fölé kellett beágyazni a még más által nem használt részekre, többnyire a C8000 kezdőcímre, de kis kapcsolókkal más kezdőcím is beállítható. A szoftver mint egy diafilmvetítő, mindig a szükséges diakockát, azaz a 64 kB-nyi területet lapozza be az ablakba, ahol a szoftver elvégezheti a szükséges adatmódosításokat, műveleteket. Az EMS bővítőkártyán maximum 32 MB RAM lehet, ennyit képes az EMS szoftverje kezelni, s ehhez 20-40 kB-nyi helyet foglal el, mint a CONFIG.SYS-be beírandó szoftvermeghajtó. Apró kitérő: nem egy szoftver vállalkozik arra, hogy az EMS memóriabővítést az annál kb. 10-20-szor lassabb merevlemezt használó szoftveremulációval pótolja (Above Disc, Turbo EMS, V-EMM).

A PC-k AT-rtékelése

A következő fejlődési lépcső a 286-os AT-k korszaka volt. Processzoruk egy speciális üzemmódjában már képes volt az 1 MB feletti memóriaterületet elérni. Ekkor terjedtek el az 1 MB-os AT alaplapok is. Komoly hibák forrása az a közkeletű tévedés, hogy ez az 1 MB az operációs rendszer szempontjából folyamatosan helyezkedik el. NEM! A Felső 384 kB mint extended memória az 100000h, azaz az 1 MB-os kezdőcímen csücsül. Az 5.0-ás verziószám előtti MS-DOS és PC-DOS (valamint a DR DOS) nem is tudta másra felhasználni, mint RAM-diszk céljaira. Az ugyancsak erre a célra írt programok (kihasználva a 286-os processzorok speciális tulajdonságát, hogy ún. protected üzemmódban képesek az 1 MB feletti memóriabővítést is elérni), már éltek ezzel a lehetőséggel, így nemcsak az adatok tölthetők az 1 MB fölé, hanem maga a program mérete is meghaladhatja a korábbi 640 kB-os korlátot. Egy protected módban működő program egy XT gépen természetesen már el sem indul. Ilyen programok például a Lotus 3.0 feletti, az AutoCAD 10.0 feletti változatai és a Microsoft Windows alapú újabb programok zöme (melyek Standard üzemmódot követelnek).

Meg kell jegyezni, hogy az 5.0 feletti DOS-változatok képesek az operációs rendszer nagyobbik részét (kernel és adatátviteli munkaterületek — BUFFERS) az 1 MB feletti első 64 kB-os részbe (HMA, high memory area) tölteni, így ezzel több mint 40 kB-ot felszabadítanak a hagyományos területről. Ezt a szolgáltatást az MS-DOS 5.0-nál a CONFIG.SYS-be írt alábbi sorokkal aktiválhatjuk:

```
DEVICE = HIMEM.SYS
DOS = HIGH
```

Mielőtt áttérnénk a 386-os gépekre, szólni kell az ún. NEAT-alaplapos (NEAT, LEAP chipset) 286-os AT-kről is, mert ezek szolgáltatásköre nagyon sok AT-286 tulajdonosának fejében okoz zavart. A NEAT-es gépek lényege: 3 évvel ezelőtt a Chips & Technologies cég kifejlesztett egy speciális áramkört, amely lehetővé teszi, hogy az alaplapon található 1 MB feletti RAM-okat az AT Setupjából a felhasználó közvetlenül extended memóriának vagy EMS-nek határozza meg. Nagyobb mennyiségű bővítés esetén (legalább 4 MB) arra is lehetőség nyílik, hogy egy időben mindkét bővítéstípus jelen legyen — természetesen megosztva. De a NEAT-es gépeknél is szükség van szoftver-EMS-driverre.

További lehetőség a NEAT-alaplapos gépeknél (figyelem, AT-286-osok közül csak a NEAT-esnél, míg a 386/486-osok mindegyikénél) az, hogy itt fizikailag is megtalálható a 640 kB és 1 MB közötti RAM-chip, s ez korlátozottan arra is felhasználható, hogy abba memóriarezidens programokat, eszközmeghajtókat töltsön fel az arra alkalmas operációs rendszer vagy az önálló memóriamenedzser program (DR DOS 5.0 és 6.0 vagy QRAM, Move'em).

Hogyan döntheti el valaki, hogy éppen egy NEAT-es AT boldog tulajdonosa-e? Kellett (volna) kapnia gépével egy olyan gyári floppyt, melyen az EMS-driverek szerepelnek, rövid leírással együtt. Ha ilyet nem kapott, figyelje a gép bejelentkezését, csípje el az AT Setupját, — ha tapasztalatlanabb, semmit ne változtasson, vagy tartson mindig kéznél egy indítólemezt —, és ellenőrizze, van-e mód az EMS-memória definiálására.

Az is előfordulhat, hogy bár erre lenne lehetőség, de lévén összesen 1 MB RAM, jobb ezt SHADOW-RAM-nak definiálni. Ez némileg meggyorsítja az operációs rendszer működését azáltal, hogy a lomhább BIOS-ROM chipek tartalmát tükrözi az azonos címen ülő, de már engedélyezett, gyorsabb elérési idejű RAM-ba. Ismét leírom a nyomatékosítás kedvéért: 286-os gép esetén csak akkor álmodozzunk arról, hogy programokat töltünk fel az UMB-be, ha NEAT-alaplapos az AT-nk!

A 386-os vagy 486-s alaplapú AT-k esetében — minimum 2 MB RAM-mal — már merőben más a helyzet. Egy NEAT-es AT-hez szokott felhasználó először talán elcsodálkozik azon, hogy nem tudja átkonfigurálni az alapértelmezésben extended memóriabővítést például EMS-szé. Nem is kell. Ezt elég rábízni a 4.0 verziószámtól kezdődő DOS-okban található EMM386 meghajtókra, de még célszerűbb egy erre kihegyezett memóriamenedzser programot használni. A legjobbak: QEMM 386 v.6.01, amely önállóan is kapható, de része a DESQview 386 v.2.4-es csomagnak is. Mindkét terméket az amerikai Quarterdeck jegyzi. A csomagokban található egy rendkívül hasznos és szemléletes tesztprogram is (Manifest v.1.11), melynek segítségével könnyen érthető, plasztikus betekintést nyerhetünk a számítógépünkben található összes memóriatároló (RAM és ROM) állapotáról, tartalmáról, sebességéről, sőt megbízható, és az adott hardverkonfigurációnak legjobban megfelelő tippeket is kapunk a PC CONFIG.SYS és AUTOEXEC.BAT állományának optimális kitöltéséről.

S ami még inkább felhasználóbaráttá teszi e szoftvereket: az önálló OPTIMIZE funkció, amely miközben háromszor is újraindítja a gépet, végeredményképpen feltérképez minden hasznosítható RAM-területet 640 kB és 1 MB között, és oda automatikusan fel is tölti mindazt, amit csak lehet (egérmeghajtót, magyar ékezetesítőt, hálózati drivereket — NET3, NET5, IPX stb.). Bárki nyugodtan rábízhatja magát a QEMM-re, vagy 286-os NEAT-alaplapos gépekhez kifejlesztett kistestvérére, a QRAM 2.0-ra. Következő számunktól kezdődően ők fognak közelebbről bemutatkozni.

Herczeg József

Közvetlen járat a floppyra

„Perszonális" programozás

A PC-boncolás során az eddigiekben megismerkedtünk a mágneslemezek fizikai felépítésével; most tekintsük át a közvetlen programozás lehetőségeit. Erről az oldalról csak a floppyegységeket tárgyaljuk, mert a winchestervezérlők sokféleségük miatt mind egyedi kezelést igényelnek.

A floppymeghajtó vezérlőegysége három i/o porton keresztül programozható. Ez a három port a következő: a 3F2 i/o címen a vezérlő digital output register (DOR) érhető el. A DOR csak írható, kiolvasására nincs lehetőség. A 3F5 i/o címen található a data register (DA). Ezt a regisztert írhatjuk, és ki is olvashatjuk belőle az utoljára beírt értéket. A 3F4 i/o címen érhető el a main status register (MSR). Az MSR csak olvasható, írására nincs lehetőség. Ez a regiszter ad információt az egyes lemezegységek aktuális állapotáról. Az innen beolvasható státuszbájt egyes bitjeinek jelentése a következő:

0. bit: az A: lemezegység foglaltságát jelzi.

1. bit: a B: lemezegység foglaltságát jelzi.

2. és 3. bit: a winchester kezelésében játszanak szerepet.

4. bit: a floppyvezérlő foglaltságát jelzi.

5. bit: ha értéke 0, akkor a vezérlő DMA segítségével mozgatja az adatokat, ha 1, akkor anélkül.

6. bit: az adatátvitel irányát határozza meg. Az 1 lemezolvasást, a 0 pedig lemezre írást jelent.

7. bit: ha értéke 1, azt jelzi, hogy a vezérlő kész az újabb parancsok fogadására.

Az MSR a vezérlő és a meghajtók pillanatnyi állapotáról ad információt. A DOR ezzel szemben lehetőséget biztosít az egyes jellemzők megváltoztatására. A DOR-ba írható bájt jelentése a következő:

0. és 1. bit: kijelöli a használandó floppyegységet.

00 az A: egységet jelöli, 01 pedig a b: egységet.

2. bit: 0 értéke alapállapotba hozza a vezérlőkártyát. Egyéb műveletek idejére ide 1 értéket kell írni.

3. bit: 1 értéke engedélyezi a DMA használatát, 0 értéke tiltja. Az IBM PC csak a DMA-módot támogatja.

4. bit: 1 értéke bekapcsolja az A: egység motorját.

5. bit: 1 értéke bekapcsolja a B: egység motorját.

6. és 7. bit: a winchester vezérlésében játszanak szerepet.

A DA regiszteren keresztül történik a floppydiszk-vezérlő programozása. A vezérlő 15+1 parancsot ismer fel. Az egyes parancsok egy parancsbájttal kezdődnek, amelyet a szükséges paramétereknek kell követnie. A parancsbájt három felső bitjének jelentése minden parancs esetében azonos. A 7. bit 0 értéke azt jelenti, hogy az adott műveletet egy szektoron kell elvégezni, 1 értéke pedig azt, hogy két átellenes szektoron. A 6. bit az információrögzítés vagy olvasás módját adja meg. 0 értéke közönséges frekvenciamodulációt jelent, 1 értéke pedig módosítottat. A DOS által használt lemezek mind módosított frekvenciamodulációval vannak írva. Az 5. bit csak olvasási műveleteknél játszik szerepet. Azt határozza meg, hogy a „write deleted data" paranccsal kiírt szektorokat a vezérlő egyszerűen átlépje-e, vagy hibajelzést adjon. A DOS nem használja ki ezt a lehetőséget, soha nem ír a lemezre ezzel a paranccsal.

A parancsbájt alsó öt bitje azonosítja a végrehajtani kívánt parancsot. Az egyes parancsok és leírásuk: (Az itt jelzett parancskód csak a parancs alsó öt bitjét szabja meg. Ehhez még hozzá kell tenni a felső három bitet, amit az előző bekezdésben leírtak alapján lehet meghatározni.)

xxx00110 — Szektor beolvasása a lemezről. A parancsbájt után még nyolc paraméterbájt kiküldése szükséges.

1. paraméterbájt: a bájt alsó két bitje a lemezegységet azonosítja (00=A:, 01=B:), a 2.bit a fej számát (0 vagy egy) adja meg. A többi bit állása közömbös.

2. paraméterbájt: a sávsorszám. Ide azt a számot kell írni, amely a szektor azonosító fejlécében szerepel (a szektorazonosító fejléc pontos felépítése az előző havi cikkben szerepel). Ez a parancs a fejet nem pozicionálja, hanem ott próbál meg olvasni, ahol éppen van. Ha az adott sávon nem talál olyan szektort, amelyben az itt megadott sávsorszám szerepel, akkor az olvasást nem hajtja végre.

3. paraméterbájt: head sorszám. Ugyanaz a szerepe, mint az előző bájtnak, csak nem a szektorsorszámra, hanem a head sorszámra vonatkozik.

4. paraméterbájt: szektorsorszám. (Ugyanúgy, mint az előző két esetben.)

5. paraméterbájt: a szektorméret jelzőbájtja. (Pontos ismertetése az előző havi részben olvasható.)

6. paraméterbájt: az adott sávban lévő szektorok száma.

7. paraméterbájt: a szektorok közti kitöltő bájtok száma. (Lásd a múlt havi részben.)

8. paraméterbájt: az átviendő bájtok száma. Csak akkor használható, ha a szektorméret jelzőbájtja 0, azaz a szektorhossz 128 bájt. (Bővebben a múlt havi részben.)

xx000101 — Adatok kiírása a lemezre. A parancsbájt után még nyolc paraméterbájt kiadása szükséges. Ezek megegyeznek az előző parancsnál leírtakkal.

xx001001 — Rejtett adatok kiírása (write deleted data). Az íly módon kiírt adatokat csak a következőként ismertetett beolvasó paranccsal lehet visszaolvasni. A parancsbájt után még nyolc paraméterbájt kiadása szükséges. Ezek megegyeznek az előző parancsnál leírtakkal.

xxx01100 — Rejtett adatok olvasása lemezről (read deleted data). A parancsbájt után még nyolc paraméterbájt kiadása szükséges. Ezek megegyeznek az előző parancsnál leírtakkal.

0xx00010 — Teljes sáv beolvasása a lemezről. A parancsbájt után még nyolc paraméterbájt kiadása szükséges.

Ezek megegyeznek az előző parancsnál leírtakkal.

0x001100 — Teljes sáv formázása. Azt a sávot formázza, amelyen a fej éppen áll. A parancsbájt után még öt paraméterbájt kiadása szükséges. 1.: megegyezik az előző parancsoknál leírtakkal. 2.: a szektorhossz azonosító bájtja. 3.: szektorok száma a sávon. 4.: szektorok közti kitöltőbájtok száma. 5.: az adatterületekre írandó bájt.

Az eddig ismertetett parancsok egy hét bájtból álló választ adnak a parancs elvégzése után. Ez a válasz informál minket arról, hogy a kívánt művelet végrehajtódott-e, vagy ha nem, akkor miért nem. Ennek a hét bájtnak a leírását az 1. táblázat tartalmazza.

00000111 — Ez a parancs visszavezérli az író/olvasó fejet a lemez fizikailag legelső sávjára. A parancsbájt után még egy paraméterbájt kiadása szükséges. Ennek alsó két bitje azonosítja a kívánt lemezegységet. A paraméterbájt többi bitje érdektelen.

00001111 — Fejpozicionálás. A parancsbájt után még két paraméterbájt kiadása szükséges. 1. paraméterbájt: megegyezik az első öt parancsnál leírtakkal. 2. paraméterbájt: a kívánt sáv sorszáma. A parancs végrehajtása után az író/olvasó fej a megadott sáv fölé áll.

Fridl György

A floppymeghajtó vezérlőegységének hétbájtos válasza a cikk szövegében jelzett parancsokra

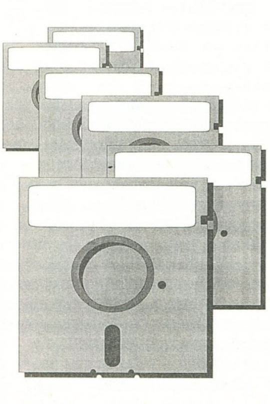
1. bájt
- 0., 1. bit: A kiválasztott lemezegység kódja
- 2. bit: A lemezoldal száma
- 3. bit: 1 értéke hibát jelez (a lemezegység nem válaszol)
- 4. bit: 1 értéke hibát jelez (mágneslemezhiba)
- 5. bit: 1 = A seek parancs hibátlanul végrehajtva
- 6., 7. bit: Hibajelző bitek, 1 értékük hibát jelez

2. bájt
- 0. bit: 1 = Szektorazonosító nem található
- 1. bit: 1= A lemez írásvédett, nem írható
- 2. bit: 1= Szektor nem található
- 3. bit: Mindig 0
- 4. bit: 1= DMA programozási hiba
- 5. bit: 1= CRC-hiba (lemezsérülés)
- 6. bit: Mindig 0
- 7. bit: 1= Szektor nem létezik

3. bájt
- 0. bit: Ugyanaz, mint a 2. bájt 1. bitje
- 1. bit: A beolvasott sávazonosító nem azonos a kívánttal
- 2., 3. bit: Az ismertetett parancsoknál nincs szerepe
- 4. bit: A beolvasott sávazonosító nem azonos a kívánttal
- 5. bit: CRC-hiba (lemezhiba) az adatrészben (hasznos részben)
- 6. bit: Rejtettadat-jelzőbit
- 7. bit: Mindig 0

4. bájt Az utolsó műveletben részt vevő sávsorszám (track)
5. bájt Az utolsó műveletben részt vevő oldalsorszám (head)
6. bájt Az utolsó műveletben részt vevő szektorsorszám (sector)
7. bájt Az utolsó műveletben részt vevő szektorméret-jelzőbájt

Lemezek gyorsmásolása



A Cédrus Kiadó Kft expressz szolgáltatása saját szoftverek, demó-programok sokszorosítására.

Másolás hozott lemezekre:

5,25" DS/DD lemezek	25 Ft/db
5,25" DS/HD lemezek	40 Ft/db

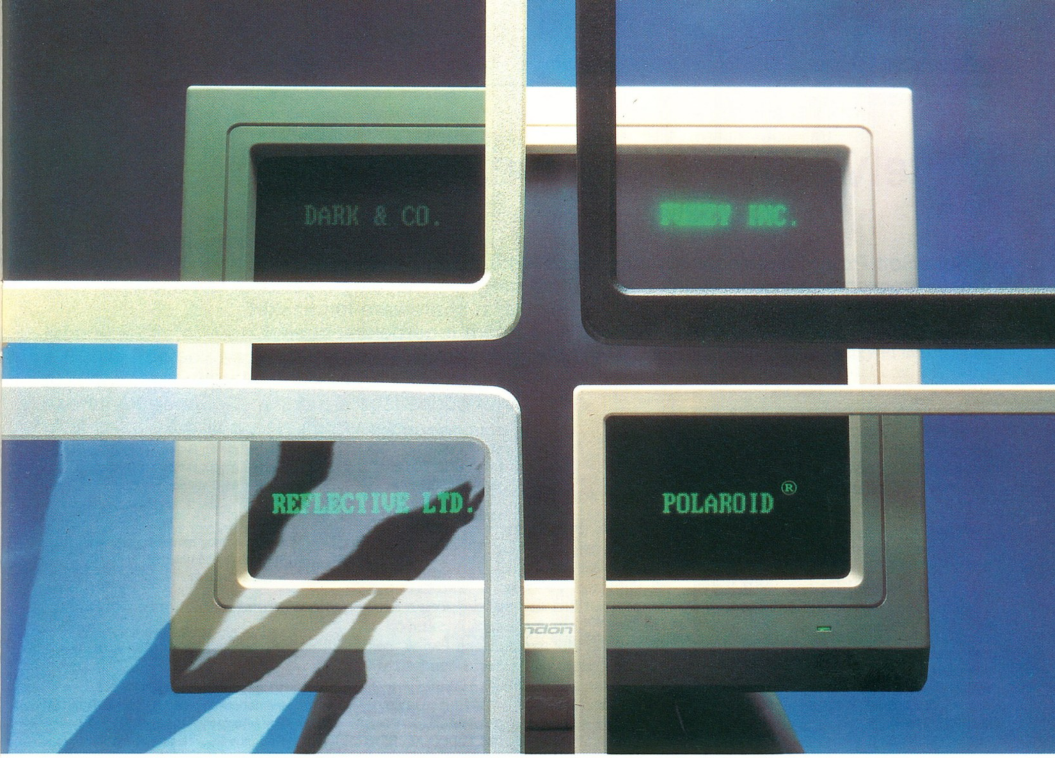
Másolás a kiadó által beszerzett lemezekre:

5,25" DS/DD lemezek	75 Ft/db
5,25" DS/HD lemezek	90 Ft/db

Címkekészítés és a lemeztasakra nyomtatás külön megállapodással.

Cédrus Kiadó Kft.
1441 Budapest VIII., Reguly Antal u. 8.
Telefon/fax: 133-1839

INFORMÁCIÓKÉRÉS: 30 ▲



Melyik szűrő felel meg az előírásoknak is, a munkatársainak is?

Aki számítógépen dolgozik, az ismeri a monitorok zavaró fényhatásait. Megerőltetett szem, nyakfájás, fáradtság. Növekvő hibaszázalék, csökkenő termelékenység.

Európában szigorúbbak lesznek a munkavédelmi előírások. A jó monitorszűrő kiválasztása fontosabb lesz, mint eddig volt.

Hogy melyiket válassza? Vannak, amelyek egyszerűen nem szüntetik meg a fényvisszaverődést. Némelyik igen, az viszont elsötétíti a betűket is. Van olyan, amelyik szétszórja a képernyő fényét, és elmosódott képet ad.



Miért éppen a Polaroid különleges körpolarizációs szűrője oldja meg a problémát? Elkápráztathatnánk Önt tudományos ismeretekkel (elvégre 50 éve finomítjuk polarizációs szűrőink technológiáját), de a legmeggyőzőbb érvek a tények: A Polaroid körpolarizációs monitorszűrő a fényvisszaverődést 99%-ban, az elektromágneses sugárzást 98%-ban kiszűri, a sztatikus feltöltődést megszünteti, a képernyő képét kontrasztossá teszi.

Polaroid monitorszűrők

Cédrus Karolina Áruház

1251 Budapest XI., Karolina út 17. Telefon: 166-2111 Telefax: 185-2221

INFORMÁCIÓKÉRÉS: 28 ▼

„Nyelvkönyvek" a hónap témájához

A SolarSoft Programkönyvtár számos olyan lemezt tartalmaz, amely programozási ismereteink bővítését, új programnyelvek elsajátítását támogatja. Nyelvenkénti csoportosításban felsoroljuk ezeket a shareware lemezeket, így mindenki könnyebben választhatja ki a számára legérdekesebbeket.

Assembler

008 ADVBAS99	ASM-bővítések Basichez
013 A86/D86 3.22, ASMWIZ	Assembler/Debugger
067 WHIZZARD SCREEN	ASM-rutinok és programok
103 ASSEMBLER PACK	Komplett assemblercsomag (10 lemezen!)
222 MAX FREEWARE EDITOR	Programszerkesztő (Unix-szerű + ASM forrás)
305 BOOSTERS	ASM-bővítések Turbo Pascalhoz

Basic

005 TURBO BASIC TOOLS	Segédprogramok Turbo Basichez
006 EDITOR IN QUICKBASIC	Szövegszerkesztő Basic forráskóddal
009 QBWARE	Basic-kiterjesztés
010 QBTOOLS & INFO	QB 2.0 kiterjesztés
011 BASWIND & BWTOOL	Ablaktechnika QB 4.x-hez
014 MS-QUICKBASIC TOOLS	Minták, bővítések (5 lemezen)
050 PDS BASE	Adatbázis-kezelés Basic-kel + oktató
087 SCREEN-DO	Képernyőtervező
127 FUNKY TOOLBOX	Basic segédrutinok
239 BASIC LEARNPROGRAM	Oktatócsomag a Basic-hez
330 TEXT & SCREEN EDITORS	Editorok Basic forrásaikkal
331 DVED SCREEN EDITOR	Editor, rajzoló, diagramkészítő
373 QBSCR SCREEN ROUTINES	QB 4.x profi képernyőkezelés
485 BASIC COMPILER	Két ragyogó fordító editorral
486 QBRTEE & QB UTILS	Segédprogramok QB 4.x-hez

C és C++

001 C-TUTORIAL	Oktatóanyag példaprogramokkal
002 TURBO C TUTORIAL	Oktatóanyag példaprogramokkal
128 C-WINDOW	Quick C és Turbo C könyvtár
169 PDVIM	Virtuális debugger
204 THE WINDOW BOSS 5.17	MSC/TC ablaktechnika
315 PERSONAL C-COMPILER	Önálló C-compiler
316 C-TASK MULTITASKING	Több feladatos futás (forrás)
334 C-WINDOW TOOLKIT	MSC/TC popup/pulldown menük
339 EXTLIB & DBSCAN	Ablaktechnika, DBF-böngésző
345 CXL C LIBRARY 5.1	MSC/TC/ZORTECH C könyvtárak
348 PC GRAPHICS C	HP BASIC grafika MSC-hez
376 STEVE'S TC LIBRARY	TC közvetlen videomemória-kezelés
396 FLASHPAC C LIBRARY	BIOS-szintű C rutinok
441 DATABASE IN C	Adatbázis-kezelés (BTREE, DBASE bázisok)
442 WINDOW PRO 1.51	Ablaktechnika (small/med/large)
443 C-MIX #1	EMS, XMS és egérkezelés C-ből
444 C-MIX #2	Tárrezidens programok készítéséhez
445 C-MIX #3	3D grafika, C fordító/LIB forrásban
446 C-MIX #4	Változó méretű RAMDISK EMS/XMS-ben
487 CSCREEN EDITOR	Programeditor C és ASM forrással
488 COMPILET TUTORIAL	Készítsünk compilert (*.C források)
495 TEGLC WINDOWS TOOLKIT	Ikongrafikus felület, ikoneditor
496 COMPLETE C C++	C++-kiterjesztés MSC/TC-hez (2 lemezen)
498 C++ TOOLKITS #1	Egér, OOP-kiterjesztések
499 C++ TOOLKITS #2	Adattömörítő program forráskódban
500 C++ TOOLKITS #3	OOP-kiterjesztések
501 C++ TOOLKITS #4	OOP-kiterjesztések
502 C++ TOOLKITS #5	OOP-kiterjesztések az #501-hez
503 TURBO C TOOLKIT	TC segédprogramok
M009 TURBO C TOOLKIT	Cache, menü, B-tree TC-hez
M010 MSC TOOLKIT	Cache, menü, B-tree MSC-hez
M042 WISE SW #1	Clipper és C rutinok forrással

Modula

003 MODULA-2 TUTOR	Oktatócsomag mintaprogramokkal
012 MODULA-2 COMPILER	Teljes fejlesztőrendszer
049 PIBTERM	Modula forrásállományokkal

Fortran

166 ACM	Fortran rutinok (matematikai)

Turbo Pascal

004 TP 5.x MULTITASK. 2.10	Több feladatos futtatás (német dokumentáció)
080 TP MODULATING	Forráslistázó
098 TURBO APPRENTICE	Rezidens fejlesztési segédeszköz
205 MINIGEN	Pascal ablaktechnika
236 TURBO PASCAL PACKET	150 forrásprogram (5 lemezen)
270 TURBO DESIGNER	TP képernyő-kódgenerátor
295 T-REF	Programdokumentálás
298 BOX SCREEN DESIGNER	Képernyőtervező forrásokkal
301 TP 4/5.0 SAMPLER	Unitok, *.PAS (6 lemezen)
302 EZ DOSS DOS-MANAGER	DOS keretrendszer (forrás!) (2 lemezen)
303 MAILPRO MAILING-SYST.	Cím-, telefonjegyzék (3 lemezen)
304 TURBO TECHNO JOCKS	Szuperunitok forrással (2 lemezen)
306 TPTC (TP TO TURBO C)	TP—TC konverter + forrás
307 PXL CROSSREFERENCE	Kereszreferencia-készítő
308 FILE MANAGER/FILESTUF	3 PAS-alkalmazás
309 LITECOMM TEL. TOOLBOX	Kommunikációs rutinok
310 MDCD	Adattömörítő
311 TP — ARC TOOLS & MORE	DEARC: PKARC-kompatibilis
312 TURBO OVERDRIVE PACK.	Lotusmenü, kalkulátor
313 TURBO ENH. TOOLKIT	Gyors videorutinok
321 TP4MENU	Menü- és helpgenerátor
322 TP4 BONUS DISK	Vegyes PAS programok
323 QWICK SCREEN UTILITIES	Szuper videorutinok
333 TURBO SPRITES	Grafikus tervezés és animáció
364 TURBO SCREDIT	Képernyőtervező TP & Turbo C-hez
370 DML & XREF	87 darab PAS rutin
408 DATABASE IN PASCAL	Adatbázis-kezelés PAS-forrással
426 CtoP 1.2b	Turbo C—TP forráskonverter!
489 PULL MENU BUILDING	Gyors menü, ablaktechnika 5.5
491 OOP #1	Objektumorientált prog. 5.5
492 OOP #2	Objektumorientált prog. 5.5
493 TP TOOLKITS	Teljes B-TREE rendszer forrásban
494 TEGLP WINDOWS TOOLKIT	Ikongrafikus felület, ikoneditor

Oktatóprogramok

001 C-TUTORIAL	Mintaprogramokkal (2 lemezen)
002 TURBO C TUTOR	Mintaprogramokkal (2 lemezen)
003 MODULA-2 TUTOR	Mintaprogramokkal (2 lemezen)
208 PASCAL LEARNPROGRAM	Általános oktató
239 BASIC LEARNPROGRAM	Teljes nyelvi leírás
300 TURBO PASCAL TUTOR	84 forrásprogrammal

Funky Toolbox

Szellemes segédprogramok

Igen sok ötletet meríthetnek a Funky Toolbox segédprogram-gyűjteményből mindazok, akik Gwbasic vagy Basica programjaikat szeretnék gördülékenyebbé, elegánsabbá tenni. (A „Funky" rövidítés a function key — funkcióbillentyű helyett áll.)

A rutingyűjtemény széles tevékenységi skálát ölel fel:

— Folytatás bármely billentyű leütésére.

— Sáv-szektor konverzió abszolút szektorrá a DEBUG-hoz.

— Hexacím konverziója decimális számmá.

— Megjeleníti a képernyőkarakterek hexatábláját.

— Hibakezelés (demo és on/off rutin).

— Mono és color üzemmódok közötti átkapcsolás.

— Nyomtatókezelés.

— Normál zene.

— Szokatlan zenei hatások.

— Háttérzene programokhoz.

— Képernyő elsötétítése.

— CPU-tesztek.

— Memóriateszt.

— Hogyan nézzünk bele DEBUG-gal állományokba/ROM-ba.

— Szöveges állományokba hogyan nézzünk bele.

— Hibaüzenetek megjelenítése F5 billentyűvel.

— Állomány titkosságának megszüntetése listázáshoz.

— Funkcióbillentyű kezelése.

— Címkézés.

A programozási példák mellett számos .DOC kiterjesztésű állományt is találunk a lemezen. Ezek egy része nagy meglepetéseket tartogat. Olyan eszmefuttatásokat olvashatunk ékes amerikai nyelven, amelyek szinte a Murphy-törvények babérjaira pályáznak. Talán legkedvesebb közülük a SENSITIVE.DOC. Itt azt fejtegetik a szerzők, hogy ki miért tart a számítógépektől. A programfejlesztőknek feltétlenül tekintettel kell lenniük a felhasználók eredendő gépiszonyára. Meg kell teremteniük a bizalom légkörét, akkor alakulhat csak ki a megfelelő ember–gép kapcsolat. Erre példának egy házasságközvetítő irodában működő rendszert hoznak fel. Itt először bemutatkozik a gép, és kezelőjének csak a szóközbillentyűt kell lenyomnia a következő képernyő megjelenítéséhez. Mivel ez általában sikerül, magabiztosabban olvassa a kérdést: „Jól látja-e a képet?" (Lehetne helyette: „Kényelmesen ül-e?") Itt sem tévedhet nagyot: Y/N válaszok közül biztosan eltalálja valamelyiket. És csak ezután kérdezik meg az új ügyfél nevét. Igaz, hogy a gépnek egyedül erre lett volna szüksége, de addigra két helyes válasz után már lelkesebben, magabiztosabban végzik el az érdemi munkát. Sok minden múlhat ilyen apróságokon!

Természetesen a .DOC állományok többsége a fenti példánál több konkrét, kézzelfogható számítástechnikai információt tartalmaz. A stílusra azonban jellemző, hogy a lemezszektorok és a memóriakiosztás precíz ismertetésekor is hol Rousseau-tól idéznek, hol pedig egy „országomat egy térképért (map)!" kitétel jelenik meg. Kellemes szórakozás a dokumentáció végigböngészése.

A Funky Toolbox a SolarSoft programkönyvtár #127-es lemezén található.

Verebély Pálné

Megduplázhatja nyomtatói számát egy újdonsággal

(És közben nem kell többé várnia a nyomtatóra)

Minden gyakorlott számítógépes szakember tudja, hogy a nyomtatás rengeteg időt pazarol el.

Még a leggyorsabb nyomtató is lassabb a legtöbb számítógépnél. Így gyakran előfordul az, hogy a számítógépnek várnia kell a nyomtatóra. Ezt az elveszett időt takaríthatja meg a Printer Manager segítségével, ugyanakkor két vagy három számítógéphez csak egy nyomtató szükséges.

A Printer Manager két fő problémát egyszerre old meg.

Az egyik funkciójában két-három nyomtatót helyettesít, a másik funkciójában intelligens memória, melyben a szövegek tárolódnak nyomtatásukig.

A nyomtatott szövegek sorbarendezve, egymás után jelennek meg.

A Printer Manager a következő kézzelfogható előnyöket kínálja az Ön számára:

1. Megtakarítja egy második nyomtató árát. Két (vagy három) számítógép dolgozhat egy nyomtatóra anélkül, hogy az adatok összekeverednének.

2. Megszabadítja a számítógépeket a várakozástól. Segítségével 4-6 perc alatt akár 1Mbyte hosszúságú szöveg is kiíródhat a Printer Manager memóriájába. A számítógép és kezelője ezután szabadon dolgozhat bármely feladaton.

Tételezzünk fel szerény 300 Ft órabért egy számítógép, és kezelője számára. Mindössze 30 perc napi nyomtatási időt számolva egy 20 munkanapos hónapban, a havi megtakarítás órákban kifejezve:

0.5[óra] • 20[nap] = 10[óra/hónap]

Évi megtakarítás Ft -ban kifejezve:

12 • 10[óra/hó] • 300[Ft/óra] = 36.000[Ft/év]

Két számítógép esetén ez az összeg megduplázódik.

3. Univerzális

Bármilyen számítógéppel dolgozhat, melynek soros, vagy Centronics portja van. (XT, AT, AT386 stb.)

Bármilyen nyomtatóval dolgozhat, amelynek soros, vagy Centronics bemenete van. (mátrixprinter, laserprinter, PostScript printer, plotter, fólia-kivágógép stb.)

4. Biztonságos

Nem fordul elő program-összeférhetetlenség, mert a működtetéséhez nincs szükség segédprogramra.

5. Megbízható

Korszerű technológia (SMT) révén 2 év cseregarancia!

6. Árak

256Kbyte memóriával 25300Ft

1Mbyte memóriával 28600Ft

4Mbyte bővítési lehetőség

Az árak az ÁFÁ-t nem tartalmazzák.

Kapható: XFER kft 1134 Budapest, Dunyov I. u. 7.

Telefon : 149-7818

INFORMÁCIÓKÉRÉS: 21 ◄

Turbo Pascal — közkincsben

A Turbo Pascal nyelvet a Borland International cég fejlesztette ki. Más Pascal nyelvjárásokkal szemben legnagyobb előnye a hatékonyság, mert a Turbo Pascal programok lefordítása igen gyorsan, egy menetben történik meg, és a kész programok futásideje is jó. Szövegszerkesztőként egy Wordstar klónt ajánlanak, ezt azonban teljesen átalakíthatjuk. A Turbo Pascal egyetlen hátrányaként szokták megemlíteni, hogy .COM kiterjesztésű állományokat hoz létre. (Kérdés, hogy ez valóban hátrány-e.)

A SolarSoft könyvtár #236 szám alatt bocsátotta ki azt az 5 lemezt megtöltő programkavalkádot, amely Turbo Pascal nyelven készült. (Igen nagy mennyiségű forrásprogram.) A teljesség igénye nélkül most felsorolunk néhányat az érdekesebb programok közül. Mivel minden egyes programhoz dokumentáció is tartozik, bárki könnyedén módosíthatja ezeket saját speciális igényeinek megfelelően.

— Labirintusszerű mintát rajzolhatunk — színesben is, ekkor szép igazán.

— Pontonként címezhető grafikát tudunk kezelni.

— Animációt készíthetünk szöveges módban. Az így kialakított képeket visszajátszhatjuk.

— A ROM grafikus táblából kiolvashatjuk a pontmintákat. Az így kapott ROM karaktermintákból készíthetünk el címeket.

— Lekérdezhetjük, hogy színes vagy monokróm monitorral dolgozunk-e.

— Zenélhetünk a zongoraprogrammal — ez felvételt és visszajátszást is tartalmaz.

— Játékprogramokhoz géppisztolyhangot állíthatunk elő. Megszüntethetjük a hangot, ha véletlenül elfelejtettük egy-egy játék végén.

— Decimális egész számot begépelve visszakapjuk ennek hexadecimális megfelelőjét. Csak adott intervallumba eső egész számot enged begépelni.

— Csak meghatározott billentyűk leütését engedélyezzük és a beolvasó rutinnal a billentyűk kódját kapjuk meg.

— Beállíthatjuk, hogy melyik legyen az alapértelmezés szerinti lemezmeghajtó. (Ennek megfordítása: az alapértelmezés lekérdezése.)

— Beállíthatjuk, illetve lekérdezhetjük az aktuális keresési utat.

— Alkönyvtárakat készíthetünk, illetve megszüntethetjük ezeket.

— Áthelyezhetünk állományokat más könyvtárba, közben át is nevezhetjük az állományokat.

— Kiköthetjük, hogy csak a lefordított programok futtatása legyen megengedett.

— Melegindítást kezdeményezhetünk.

— Rezidens programmal visszahozhatjuk az utolsó tíz DOS parancsot.

— Paramétereket adhatunk át programoknak a DOS parancssorból.

— Beolvashatunk max. 80 karakteres paramétersztringeket.

Találunk továbbá telekommunikációs programot, aszinkron kommunikációs rutingyűjteményt, soros kommunikációs rutinokat, kereszteferencia-listázót, jelszókezelő programot is.

Verebély Pálné

Amitől a C gyorsabb, szebb, kényelmesebb

A FlashPac C könyvtár a Microsoft és a Turbo C nyelvű programoknál használható DOS-környezetben. A benne lévő alacsony szintű könyvtári rutinok lehetővé teszik a programozók számára a képernyő, a nyomtató, a lemez és az egér kezelését. A rutinok nem azzal a céllal készültek, hogy helyettesítsék a C nyelv szabványos lehetőségeit, hanem hogy kényelmesebb programozást biztosítsanak DOS-környezetben. E könyvtár felhasználásával gyorsabb és szebb C programokat írhatunk.

A könnyebb kezelhetőség érdekében a globális változók számát a szerzők minimálisra csökkentették. Az összes rutin Assembly nyelven készült, a Pascal nyelv paraméterátadási konvencióit használva.

A lemezes műveletekkel kapcsolatos rutinok számos olyan DOS függvényt váltanak ki, amelyek a lemezes állományok kezelésekor bájtsorozatokkal dolgoznak. Közvetlenül elérhetjük a képernyőt, tartalmát elmenthetjük, visszaállíthatjuk, ablakokat keretezhetünk, vezérelhetjük a kurzort. Az alapvető egérkezelés mellett találunk egy eseménykezelőt, BIOS nyomtatófüggvényeket, továbbá DOS és BIOS billentyűzetkezelő függvényeket.

(A Flashpac C Library a SolarSoft #396 lemezén található.)

DATENTECHNIK

Kereskedelmi Képviselet Telefon: 175-0182
Budapest I. Naphegy tér 8.
Digitális átviteltechnika, speciális távközlési rendszerek, integrált hangú és adatcsatornás készülékek (DOV), professzionális modemek nagy választékban (WAN) postai engedéllyel, csomagkapcsolás, Ethernet-Token Ring helyi hálózatok (LAN), különböző hálózatok integrálása, minőségi PC-k.Komplett hálózatok tervezése, telepítése, egyedi megoldások.

Telefax: 175-0182
Telex: 22-4800

INFORMÁCIÓKÉRÉS: 19 ◀

A Compack 4.4 csomag

Tömörített csemege

A tömörítőprogramok nemzetközi kínálata nemrégiben egy újabb használható csomaggal bővült. Egy olasz programozó (W. J. Collis) megmutatta, akárcsak Yoshi az LHA-val, Sawatzki és Nischke a Hyperrel és a francia Bellard az LZEXE-vel, hogy nem csak Amerikában lehet szupertömörítő programot írni. A Compack 4.4 csomag hasznos eszköz mindenki számára, aki lemezein helyproblémával küszködik, vagy nehezebben visszafejthetővé akarja tenni saját fejlesztésű programjait.

A Compack egy CPK44.EXE (a SolarSoft #514-es lemezen CPK44#.EXE) nevű önkibontó állományban kerül forgalomba, ezt a csomag részét képező BUILDSFX segédprogrammal hozták létre. A programcsomag telepítése mindössze annyiból áll, hogy egy (lehetőleg pathon levő) könyvtárba lépünk be, és onnan indítjuk el ezt a CPK44.EXE állományt. Az SFX kibontja magából mind a hat belepakolt állományt (az esetleg meglevő azonos nevűeket felülírja). Használatbavétel előtt érdemes átfutni a README és a MANUAL.DOC állományok tartalmát. Január közepéig elkészül a magyar nyelvű leírás is.

A Compack és a BUILDSFX a Prominence Computer Services Ltd. szerzői joggal védett programjai, de shareware programként terjeszthetők. A programot módosítani tilos. Ha elégedettek a programmal, küldjenek 25 USD-t. Aki 40 USD-t vagy többet küld, bejegyzett felhasználóvá válik, és elküldik neki a legfrissebb verziót egy nyomtatott kézikönyvvel.

Ha a Compackot vagy a vele tömörített programokat kereskedelmi célokra vagy állami intézményeknél használják, a regisztráció kötelező. A Compackkal tömörített programok terjesztéséhez írásbeli szerződés szükséges (kivéve a public domain és shareware programokat, ha a Compack adatait feltüntetik).

A kereskedelmi felhasználók, akik a Compack és a BUILDSFX segítségével kívánnak programokat terjeszteni, lépjenek érintkezésbe további adatokért a Prominence Computer Services Ltd. irodájával. A mintaszerződés megtalálható a kézikönyvben.

Bár rengeteg tömörítő csomag van, amely nagy hatékonysággal zsugorítja állományainkat, ezek egymással rendszerint nem kompatibilisak, és az archív fájl kibontásához szükség van egy kibontó segédprogramra, vagy többnyire tekintélyes méretű (5-16 kbájtnyi), az archívval egybeépített önkibontó (SFX) fejre. Amikor valakinek csak arra lenne szüksége, hogy az SFX csomagot kinyissa, az nem nagyon méltányolja a tömörítők többségének extraszolgáltatásait. Ezeket a minimális funkciókat biztosító SFX-et pedig legegyszerűbben és legtömörebben a Compack csomaggal készíthetünk.

A COMPACK.EXE nem egy archiválóprogram, mint az általános fájltömörítők, hanem az LZEXE és PKLITE programokhoz hasonlóan csak az EXE és COM állományokat zsugorítja össze a memóriában az indításkor eredeti formájában kinyíló állománnyá. Ez egyben meghatározza a használhatóság korlátait is.

Kevés kivételtől eltekintve jobban tömörít, mint a legjobb tömörítő programok, és mindössze néhány száz bájtnyi fejet és farkat ragaszt a tömörített állomány elejére és végére.

Használata:

compack \ path \ program.be \ path\ program.ki [üzenet] [–kapcsolók]

Sem a bemeneti, sem a kimeneti állomány nevét nem lehet dzsóker (* és ?) karakterekkel megadni, egyértelmű nevet kell beírnunk. A kapcsolók (r, o, i, g, h) mindig egy bevezető – (mínusz) jel után következhetnek, ismertetésük pedig angol nyelven előhívható a Compack paraméterek nélküli indításával. Normál esetben a Compack olyankor nem tömörít, ha az elérhető helynyereség kisebb 1 kbájtnál, de az –r kapcsolóval ilyenkor is pakolásra késztethető (ezáltal jelentősen megnehezítve a Compackkal tömörített program visszafejtését). Az –i kapcsolót használva a már tömörített programokkal nem vacakol. A – g mintegy kétszeresére gyorsítja a program működési sebességét, miközben átlagban csak 1%-kal lesz nagyobb a zsugorított program, mint – g nélkül. Az – o a megrendelési formanyomtatványt hozza a képernyőre, ami a Print Screen segítségével nyomtatható. A – h egyes EXE fájlok tömörítését javíthatja.

Ha macskakörmök (") között megadunk bármiféle üzenetet, az beépül a tömörített állomány végére, és kiválóan megfelel a sorozatszám, illetve a regisztrációs adatok beültetésére.

Ajánlható meghívások:

Compack bigprog.exe –g fastprog.exe

for %q in (*.com) do Compack %q d:%q –r

for %q in (*.exe) do Compack %q d:%q „Copyright (c) Fox 1991"

A Compack és a tömörített COM és EXE fájlok hardver- és szoftverigénye minimálisan 8088-as CPU, 2.0 feletti DOS, 256 kilobájt szabad RAM.

A BUILDSFX olyan segédprogram, amely a dzsóker (* és ?) karakterekkel is megadható nem túl nagy (a memóriában elférő) méretű állománycsomagot először egy EXE állományba pakolja össze, majd ennek tömörítésére meghívja a Compack programot, amelynek pathon kell lennie.

Használata:

BUILDSFX forrásfájl_maszk kimeneti_program.EXE

Mind a forrásfájl, mind a kimeneti program megadásakor patht is megadhatunk. Ramdrive esetében ajánlatos egy alkönyvtárban dolgozni, mert a lemeznév a program Ramdrive gyökerében megoldhatatlan problémát okoz. Működéséhez 2.0 vagy újabb DOS és 256 k szabad RAM kell.

Példa:

BUILDSFX D:\TXT*.DOC E:MAKEDOC.EXE

A csomag harmadik programja, a COM2EXE a pongyolára sikeredett COM fájlokat, az úgynevezett multiszegmens COM fájlokat alakíthatja vissza EXE formába, amit a Compack alaposan tömöríthet.

Használata:

COM2EXE program.com program.exe

A REPACK.BAT egy olyan batchfájl, amely megnézi, hogy a megadott forrásprogram tömörített-e, és ha igen, külső segédprogramok igénybevételével újracsomagolja, a Compackot használva.

Nagy Gábor

Jön, jön, jön... és már itt is van!

Biblia

A Biblia teljes szövegű adatbázisát készítették el az Arcanum BT munkatársai IBM PC gépen használható formában. Eredetileg CD-re készült a program, de átdolgozták 5,25"-os lemezeken terjeszthető formára is. A feladat nehézségére jellemző, hogy a csomag a jelenlegi legjobb tömörítővel, az ARJ-vel tömörítve is csak 5 db HD (1,2 MB-os) lemezen fér el. A shareware-változatba ennek egyik lemezét dolgozták bele, amely értelemszerűen szűkebb a teljes verziónál, de már önállóan is használható (SolarSoft #M050). Iskoláknak és PC-vel rendelkező szülőknek ajánljuk.

Tetris

A már meglevő Tetris-változatok népes családját egy újabbal bővítették ki. Aki szereti ezt a fajta játékot, nem csalódik. (SolarSoft #515)

Autóelszámolás

A Pneumatika BT Bosch-képviselet támogatásával, BAUTO.EXE néven született meg a SolarSoft #M051 lemezen megtalálható, Clipper '87-ben készített public domain jelleggel terjeszthető autóelszámolási program. Semminemű korlátozást nem tartalmaz (!), teljesen funkcionális, magyar nyelvű segítő képernyőkkel működik. Akinek megtetszik, használja és adja tovább! A bejegyzett felhasználóknak szükség esetén egyedi igényeiknek megfelelően módosíthatja a programot. A program fejlesztőjének neve, címe és telefonszáma megtalálható a programban. Híreink szerint a Pneumatika BT a közeljövőben újabb shareware programok kifejlesztését is szponzorálja.

GS-Auftrag

A Németországban több tízezer példányban bejegyzett GS programcsalád egyik első tagját üdvözölhetjük hazánkban. A német cím ne ijesszen el senkit, a program 1.61-es verziója teljesen magyarítva kerül a polcra. Akinek megnyeri a tetszését a shareware-változat, nem kell külföldön regisztráltatnia a programot, jó magyar forintért is megteheti a programcsalád magyarországi fordítójánál és terjesztőjénél, Kertész Zoltánnál. A lemezen természetesen a program leírása is magyarul olvasható, már a shareware-verzióban is, ezért a lemez a SolarSoft magyar szekciójában kapott helyet (#M052). A terjesztő a programcsalád további tagjait is „megmagyarítja", ami ez esetben nemcsak egyszerű fordítást jelent, hanem hogy a programokat a magyar viszonyokhoz is hozzáigazítja. A szoftver igen alkalmas az értékesítéssel kapcsolatos komplett műveletek követésére és dokumentálására, az ajánlattételtől kezdve a szállítólevélen át a számlázásig és a jóváírásig. Nyilvántarthatjuk vele a vevőket, a raktárkészletet, a követeléseket, a forgalom alakulását hálózatos környezetben is.

1FK

A Pankotai Állami Gazdaság (Szentes) programozói készítették az 1FK nevű főkönyvi könyvelési rendszert. A program villámgyorsan dolgozik, magyar nyelvű helpeket ad, és könnyen megtanulható. A shareware-változat (SolarSoft #M053) egy némileg korlátozott demó, amelynek kipróbálása után valószínűleg sokan megveszik a teljes verziót is. A bejegyzett felhasználók a teljes verzióval megkapják a nyomtatott kézikönyvet, továbbá a szoftver későbbi továbbfejlesztésekor az új változatokat. A program kézikönyve a Floppyland szakboltban megtekinthető.

Antivir-2

Hosszú ideje gyűjtögetjük a különböző magyar fejlesztésű víruskereső és -irtó programokat. Ezek közül Leitold Ferenc CHKVIR programjának csak kereső funkcióval rendelkező verziója már bekerült #M048 számmal a SolarSoft lemezek közé, az #M054 számú lemezen pedig hamarosan közreadjuk a többi beérkezett programot, mielőtt még aktualitásukat vesztik.

TTL IC katalógus

Pete László szolnoki programozó munkája a SolarSoft #055 lemezre kerülő TTK IC katalógus. A program és a katalógus elsősorban oktatási célokra hasznos, de mindenkinek ajánlható, aki áramköröket tervez TTL IC-k felhasználásával. A program C-ben készült, adatállománya viszont közönséges ASCII szövegállomány, tetszőleges szövegszerkesztővel bővíthető. A katalógus shareware verziójának adatállománya nem teljes, a teljes verzió a szerzőnél 499 forintért (plusz postaköltség) megvásárolható. A programból nyomtatni is lehet.

Memory Master

Az angol PC Plus magazin lemezmellékletén találtunk egy ügyes memóriafejlesztő programot. Ezt a csomagot a SolarSoft #516-os számú lemezen találhatják meg az érdeklődők.

Előzetes a további tervekről

Először is, amint azt ígértük, a SolarSoft katalógust havonta fogjuk frissíteni, és január közepe-vége tájára elkészül a témák szerinti programvisszakeresést is lehetővé tevő modul. A csekély érdeklődést kiváltó programokat, amelyekből másolószolgálatunknak az utóbbi fél évben nem kellett pótlásokat készítenie, visszavonjuk, de átmeneti ideig, egy-egy példányt készenlétben tartunk.

Terveinkben szerepel a már meglevő programok újabb verzióinak beszerzése és a lemezek felfrissítése, és ezt az akciót ki szeretnénk terjeszteni a mostanra kissé elavult oktatócsomagokra is. Hamarosan felújítjuk hipertext- és helpkészítő programjaink lemezeit is.

Jelentős változás, hogy egységesen az ARJ tömörítő program használatára állunk át, amint sikerül beszereznünk annak „Security Envelope" változatát a programkönyvtár karbantartásához. Ez vásárlóink számára — és a mi számunkra is — nagyobb biztonságot jelent, mert az ezzel összecsomagolt fájlokat az ARJ egyre elterjedtebb shareware-változatával csak kibontani, nyomtatni vagy tesztelni lehet, módosítani viszont nem. Az átcsomagolás elvégezhető ugyan az ARJ shareware-változatával is, ebben az esetben viszont kibontáskor nem jelenik meg az eredetiséget jelző üzenet. A fenti módosításokat abból a célból vezetjük majd be, hogy teljességgel kizárhassuk a véletlen vagy szándékos vírusfertőzést és az eredeti SolarSoft lemezek integritásának megzavarását.

Nagy Gábor

It's CeBIT Time

Izgalmas világra szóló újdonságok – csak Hannoverben!

A CeBIT '92 vásáron mutatnak be a nagyközönségnek „élő", a gyakorlati alkalmazásnak megfelelő környezetben, újonnan kifejlesztett termékeket.

Hogy hol tart ma a világ információs és kommunikációs technikában, azt 5000 kiállító, 45 országból, a teljes szakmai skálát átfogva vonultatja fel.

Exkluziv információk a már megszokottan jól informált szakmai körtől, az esemény helyszínéről.

HANNOVER
MÁRCIUS 11–18, 1992

CeBIT

Világközpont • Irodatechnika • Információ • Telekommunikáció

DEUTSCHE MESSE AG, HANNOVER/GERMANY

További felvilágositást ad: 'presentex Vertretungs GmbH Budapest, Vásárközpont, Albertirsai út 10, Levélcím: 1475 Budapest, Pf. 291, Tel.: (01) 157-4280, Fax: (01) 163-2605

INFORMÁCIÓKÉRÉS: 16 ▼

Lebegő matematika II.

Koprocesszor-specialitások

A matematikai processzorok szabványosítási folyamatában a speciális értékek előírásai is igen jelentősek.

A 0 ábrázolásánál mind a mantissza, mind a kitevő minden bitje 0, az előjel szerint megkülönböztethetünk pozitív és negatív zérót. Nulla érték keletkezik, ha az eredmény annyira kicsi, hogy egyetlen értékes jegye sem ábrázolható (alulcsordulás).

Ha a kapott eredmény normál alakban nem ábrázolható, mert ez a kitevő alulcsordulását okozná, denormált operandust kapunk. Ekkor az eredmény használható, de mivel a mantissza elején 0-ák vannak, pontossága kisebb, mint amit a mantissza hossza lehetővé tesz.

Túlcsordulás esetén — ha a szám értéke meghaladja a legnagyobb ábrázolhatót — a végtelent reprezentáló eredményt kapjuk, amelynél a kitevő minden bitje 1, a mantissza pedig 100..00. A végtelen is lehet pozitív vagy negatív.

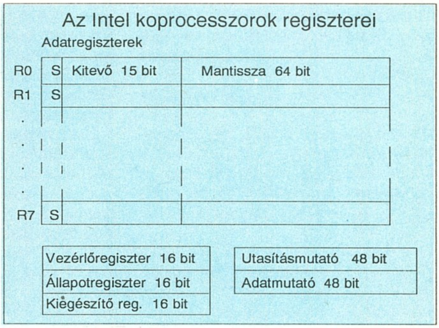
A szabvány definiálja még a nem szám (not a number = NaN) típust is, amellyel általában a beállítatlan (uninitialized) értéket jelzik. Nem szám az operandus, amelynek kitevője 11..11, mantisszája pedig nem 100..00.

Az Intel matematikai processzorai

A matematikai processzorok a számokat normál alakban tárolják, és több magasabb rendű függvényt hardverúton számítanak ki, a beépített mikroprogram felhasználásával. Ez magyarázza magas árukat.

Ezeket az eszközöket a PC-k előtti időkben a CPU egyszerű perifériaként kezelte. Egy regiszterbe kellett írni az operandust, egy másikba a parancs kódját, és bizonyos idő múlva szintén egy regiszterből kiolvasni az eredményt. Egy áramkör így többféle processzor mellett is alkalmazható volt, az Intel a 8232-t ajánlotta a 8080/8085-ös, 8 biteseihez.

A cég első 16 bites processzorainak kifejlesztésekor új, addig csak a nagygépek köréből szokásos elv szerint külön processzort készített a perifériaműveletek és a lebegőpontos számítások támogatására. Ezek önálló működésre nem képesek, de együttműködve a CPU-val átveszik feladatainak egy részét, így növelik a rendszer teljesítményét. Működési elvük miatt kapták a koprocesszor vagy társprocesszor nevet.



2. ábra

A perifériaműveleteket támogató 8089-es nem terjedt el, a 8087 típusszámú matematikai koprocesszort azonban az IBM beépítette az első PC-be. Az Intel pedig a 80286-os és a 80386-os CPU-k támogatására továbbfejlesztette a 8087-et is, így jelent meg a 80287 és a 80387. A 80486-os processzor pedig már beépítve tartalmazza a 80387-et.

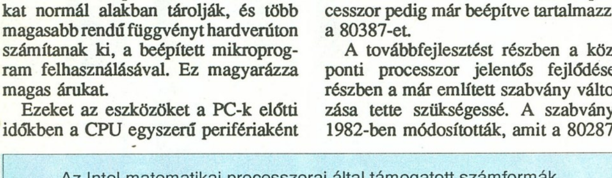
A továbbfejlesztést részben a központi processzor jelentős fejlődése, részben a már említett szabvány változása tette szükségessé. A szabványt 1982-ben módosították, amit a 80287-tel követtek. A jelenlegi formát 1985-ben adták ki, ehhez legjobban a 80387 alkalmazkodik. A koprocesszorok fejlesztése során ügyeltek a szoftverkompatibilitásra, a programok változtatás nélkül futnak az újabb típusokon is.

Az Intel matematikai processzorai által támogatott adatformátumokat az 1. ábra mutatja, a 2. ábrán pedig a regiszterek láthatók. Az operandusok tárolására 7 regiszterből álló tömböt használhatunk. A regiszterek kialakítása megfelel a normál alakú számábrázolásnak, a mantissza és a kitevő hossza pedig lényegesen meghaladja a szabványban előírtat, így biztosan teljesítik a pontossági követelményeket. Az egyes adattípusok a memóriában az 1. ábrán látható formában helyezkednek el, a processzor betöltéskor automatikusan a regiszterek — nagyobb pontosságú — alakjára konvertálja őket, az eredmény pedig — a programozó felelősségére — szintén bármelyik formában a memóriába küldhető.

A koprocesszor működését a 3. ábrán részletezett vezérlőregiszter egyes bitjeinek állításával befolyásolhatjuk, hatásukról pedig az állapotregiszter (4. ábra) ad információt.

Az egyes bitek vizsgálata előtt meg kell ismerkednünk a kizárással (exception).

A CPU által kezelt eszközökök, ha végeztek a feladattal (esetleg hibát észleltek), megszakítással kérhettek figyelmet a CPU-tól. Ha a processzor a



1. ábra

megszakítást valamilyen okból nem fogadja, az eszköz általában várakozni kényszerül. A 8087-es szintén megszakítást kér a CPU-tól, de ha az ezt nem fogadja, a matematikai processzor előre meghatározott módon folytatja a működését, ami hibás eredményre vezethet. (Ez a programozó gondja.) Az ilyen típusú megszakítást nevezik kizárásnak.

A vezérlőregiszter alsó 6 bitje azt jelzi a koprocesszornak, hogy a programozó milyen esetekben engedélyezi és melyekben tiltja a CPU-nak szóló megszakítást. Az állapotregiszter ugyanilyen nevű bitjéből tudja meg a programozó, hogy a koprocesszor a többféle lehetőség közül pontosan miért kért megszakítást (ha az engedélyezett volt), illetve kért volna, de nem volt engedélyezve.

A vezérlő- és állapotjelző biteket követve jobban megérthetjük a koprocesszor működését és lehetőségeit:

Érvénytelen művelet jelzése

Tipikus példája a negatív számból való gyökvonás, vagy ha olyan regiszterrel akarunk műveletet végezni, amelybe nem töltöttünk operandust.

Denormált operandus

A művelet denormált operandust használt.

Osztás nullával

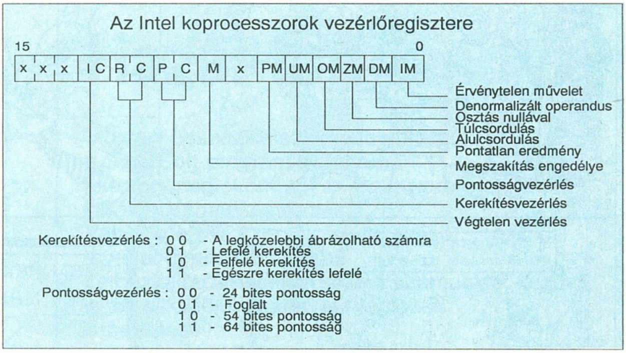
Ha a kizárás tiltott, eredményül végtelent kapunk.

Túlcsordulás

A kapott erdmény túl nagy, a regiszterekben nem ábrázolható, a koprocesszor végtelent ad eredményül.

Alulcsordulás

Az eredmény túl kicsi, denormált számot kapunk, vagy extrém esetben nullát.



3. ábra

Pontatlan eredmény

A kapott eredmény két ábrázolható szám közé esett, a koprocesszornak kerekítenie kellett.

Megszakítás engedélyezése/tiltása

Minden előző megszakítást együtt tilthat. (A 80287 és a 80387 már nem használja.)

Pontosságvezérlés

A koprocesszor nem a regiszterek által meghatározott maximális, hanem a szabványban előírt hosszúságú formákra kerekít. Ezt bizonyos fordítók és más, szintén a szabványhoz alkalmazkodó matematikai processzorokkal kompatibilis alkalmazások igényelhetik. A pontosság csökkentése nem növeli a számítások sebességét.

Kerekítésvezérlés

A normál alakú számábrázolásnál a kerekítés gyakran szükséges. A koprocesszor nem csak akkor kerekít, ha az eredmény nem ábrázolható pontosan. Mivel a saját regiszterei nagyobb belső pontosságot tesznek lehetővé, mint a kezelt adattípusok, mikor az eredményt kiírjuk a memóriába, szintén kerekítenie kell. Beállíthatjuk a felfelé vagy lefelé kerekítést, kerekíthetünk a legközelebbi ábrázolható számra stb.

Végtelen vezérlés

Meghatározza, hogy a végtelen eredmény kezelésénél figyelembe vegye-e az előjelét (affine mód) vagy sem (projektív mód). A 387-es koprocesszor — a szabvány módosításának megfelelően — csak az affine módot használja.

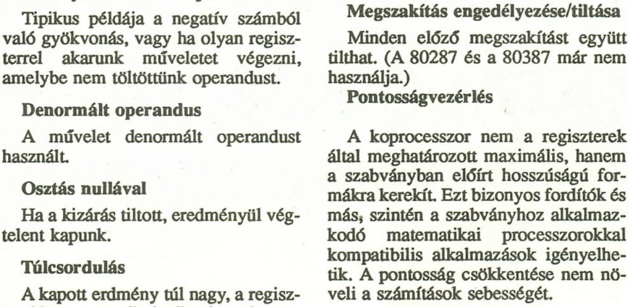
A 4. ábrán látható állapotregiszter alsó bitjei a különböző típusú kizárásokat jelzik. A feltételbitekből lehet kiolvasni az összehasonlító utasítások eredményét, értelmezésük a használt utasítástól függ.

A regisztertömböt a koprocesszor veremként kezeli, az egyes regiszterek számozása — 0-tól 7-ig — pedig a verem legfelső regiszterének helyzetétől függ. Ha tudni akarja, hogy éppen melyik regiszter a verem teteje, akkor ezt az ST bitek jelzik az állapotregiszterben. Végül a foglalt bit azt mutatja, hogy a koprocesszor dolgozik, az előzőleg kapott utasítás végrehajtását még nem fejezte be.

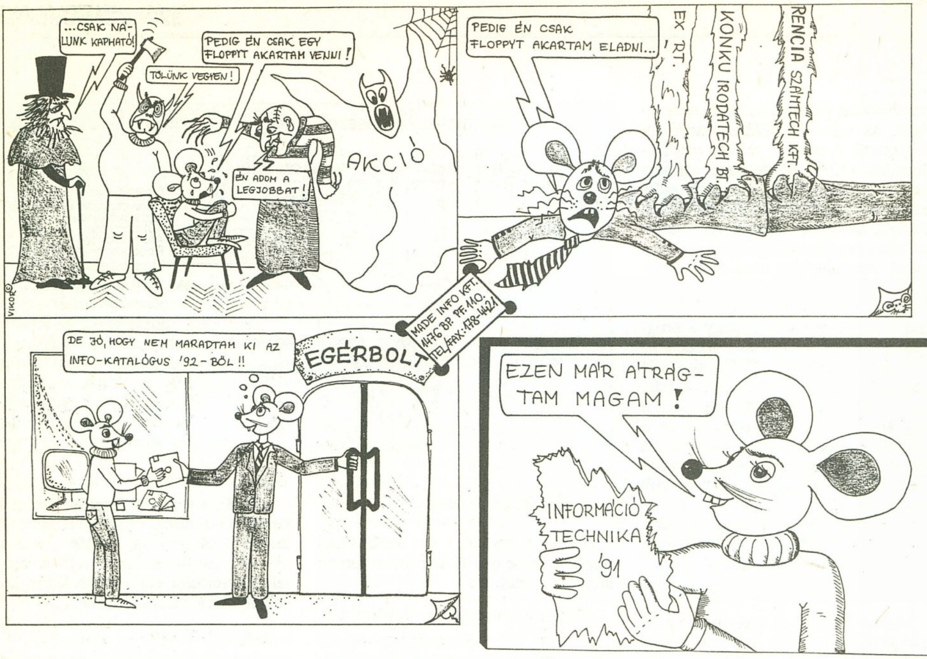
A kiegészítő (tag) regiszterben minden adatregiszterhez tartozik két bit, amelyek az egyes regiszterek tartalmáról adnak információt. Ezeket a biteket a processzor állítja be, a programozó csak olvashatja.

Az utasítás- és adatmutató regiszterek az éppen végrehajtott utasítás és az utoljára betöltött operandus memóriacímét tartalmazzák. Ha valamilyen hiba — kizárás — lép fel, kiolvasásukkal megállapítható, hogy milyen utasítás és operandus okozta a kizárást.

Csórián Sándor



4. ábra



INFO KATALÓGUS '92

EGY HELY, AHOL MEGMUTATHATJA

A KONKURENCIÁNAK,

HOGY MIBEN KÜLÖNB,

A FELHASZNÁLÓNAK,

HOGY MIÉRT PONT ÖNTŐL,

ÉS ÖNMAGÁNAK,

HOGY VERSENYBEN MARAD!

IRODATECHNIKA

Ami nélkülözhetetlen,
ami visszahívásra figyelmeztet,
ami írott üzenetet továbbít,
ami akár négy színnel is ír,
ami színeset is másol, kicsinyít,
szortíroz, nagyít.

És amiről még ma sem tudom,
hogy holnap már azzal dolgozom.

MUTASSA BE!

IRODABÚTOR

Kényelmesebb?
Modernebb?
Kis helyen is praktikus?
Mindennek van helye?
Elegáns?
Az emberhez tervezték?

MUTASSA BE!

HARDVER

Ha gyorsabb,
ha kisebb,
ha könnyebb,
ha megbízhatóbb,
ha modulárisabb,
ha olcsóbb,
ha nemcsak beszél, hanem ért is
ha már maga sem hiszi,
hogy ilyen is létezhet, akkor...

MUTASSA BE!

SZOFTVER

Ez időt takarít meg,
ez megoldásokat mutat,
ez nyilvántartást vezet,
ez kiadványt szerkeszt,
ez helyettem is számol
és számláz, rendszerez,
ez az életemet is egyszerűbbé
teszi.

Ezt még nem ismerem!

MUTASSA BE!

TÁVKÖZLÉS

Kis központ?
Több fővonal?
Sok mellékállomás?
Sokmemóriás?
Üzenetrögzítős
De ha CHIP kártyás, ha mindig
kéznél lehet,
akkor is

MUTASSA BE!

EGYÉB

Oktat?
Szervez?
Szolgáltat?
Kereskedik?
Egy húzásra azonosít?
Az értéket megóvja?
Hogyan?

MUTASSA BE!

MADE–INFO KFT. 1476 BUDAPEST, PF. 110 TELEFON: 178-4421 TELEFAX: 178-4421

INFORMÁCIÓKÉRÉS: 26 ▲

IBM PC

Hazai termés IV.

Mostani összeállításunkkal befejezzük a SolarSoft programkönyvtárban jelenleg meglévő magyar shareware-programok katalógusának közreadását. Várjuk a hazai szerzők jelentkezését további új programjaikkal! Lemezkalauzunkat következő számunktól kezdve ismét a külföldi shareware-lemezek ismertetésével folytatjuk.

SOLARSOFT KATALÓGUS

A programok ára: lemezenként 399,- Ft + áfa

Értékesítés:
FLOPPYLAND
Budapest V., Váci u. 84.
Telefon/Fax: 118-2651

Lemezszám: **M037**

Név: **Postaforgalom**

Szerző: Keszte György

Leírás: A program kis- és közepes méretű vállalatok (50-70 ezer irat/év) ki- és bemenő postaforgalmának kezelésére készült. (Alkalmazási referencia: Székesfehérvári Könnyűfémmű.) Kiszolgálja a vállalathoz beérkező és onnan kifelé irányuló postafofgalom adminisztrációjával kapcsolatos összes tevékenységet:

— A korrekt iktatókönyv vezetését.
— A postaküldemények feladójegyzékének elkészítését.
— A postaküldemények szervezeti egységek szerinti szignálását.
— Az iratok gyors visszakeresését.
— A felhasználó által definiálható listák készítését.
— A napi postaforgalom naplózását.
— Az éves szintű adatarchiválást.

Állományok:
— INSTALL.BAT
Üzembe helyezés merevlemezre
— M000.DOK
A SolarSoft könyvtár adatlapja
— REGISTER.DOK
Regisztrációs lap a fejlesztőnek
— SOLAR.DOK
Nyilatkozat
— PF.BAT
Programindító batch-fájl
— PF1.EXE
Programfájl
— PF.HLP
Help szövegfájl
— ERKEZO.DBF
Adatállomány
— KIMENO.DBF
Adatállomány
— E_WORK.DBF
Munkaállomány
— K_WORK.DBF
Munkaállomány
— IKTATO.DBF
Iktatóhelyek, adatállomány
— DRIVES.DRV
Környezetleíró fájl
— E_NAPLO.FRM
Formátumfájl
— K_NAPLO.FRM
Formátumfájl
— KEYTECH.COM
Cserélhető ékezetes billentyűdefiníció

Lemezszám: **M038**

Név: **Mandelbrot**

Szerző: Molnár Ferenc

Leírás: A program Mandelbrot halmazokat állít elő. Könnyen kezelhető és a C nyelvet egy érdekes feladat megoldásán keresztül még jobban megismerteti. A teljes C forráskód rendelkezésre áll(!), így a továbbfejlesztésre is van lehetőség.

A program néhány jellemző paramétere:
— Forráskód: Turbo C 2.0

— Forráslista:
mbe.c (52848 bájt, 1777 sor)
— Futtatható programkód:
mbe.exe (74614 bájt)
— Szükséges segédfájlok:
egavga.bgi (5363 bájt)
goth.chr (8560 bájt)
mbe_0000.scr (112000 bájt, tömörített alapkép)
mbe_0000.spk (171 bájt, alapkép koordinátái)
(Ha az mbe_0000.scr és az mbe_0000.spk fájl hiányozik, a program újra előállítja azokat.)
— EGA képfelbontás (640 x 350)
— Egy AT 386-os géppel (25 MHz) az alapkép 1 perc 56 másodperc alatt készül el.
— A képgenerálások időtartamát is jelzi, így a program egyfajta sebességmérőként is felhasználható.
— A generált képek lemezre menthetők, illetve onnan beolvashatók.
— A képek 10 színkombinációban készíthetők el. A 10 színkombináció folyamatosan változtatható, ezáltal lüktető, figyelemfelkeltő kép állítható elő (például egy kiállítási standon).
— A program a tömörített képmentés miatt a paletta 10 színét használja (egy kép = 112000 + 171 bájt).
— A halmaz megfelelő részletének kiválasztása egy ablakkal vagy közvetlen koordinátabeírással történhet.
— Az aktuális ablakkoordináták közvetlenül leolvashatók.
— A kép tetszőleges részéről ASCII szövegfájl készíthető, ami az adott részletet tartalmazza (mint egy gobelin-minta). Egy képpont = egy karakter (a karakter a képpont színére utal).

Lemezszám: M039

Név: GIB_DEMO

Szerző: Scriptum Kft.

Leírás: Ezen a lemezen a Magay—Kiss: Angol—magyar, magyar—angol zsebszótár számítógépes változatának demója található, amely mindkét szótárból csak az a-val kezdődő címszavakat tartalmazza.
A program indítása: GIB_DEMO.
A demonstrációs példány szabadon másolható. A regiszrált változat másolásvédett (lásd #M056).
A regisztrált verzió beszerezhető a szerzőknél (Scriptum Kft. 6771 Szeged-Szöreg, Pf. 2.), továbbá a Cédrus Karolina Áruházban vagy a Floppyland szakboltban 5000 Ft-os áron.

Lemezszám: M040

Név: Great Speculator

Szerző: Czél István

Leírás: Grafikával támogatott interaktív üzleti táblázatkezelő és nyilvántartó program, amely a Sunsoft Gmk támogatásával készült 1990-91-ben, üzleti, valamint tudományos célokra.
A lemezen a program demó verziója található egy önkicsomagoló állományban. A kibontott program mintegy 800 K helyet foglal el. Első elindítása előtt (amit a TDEINDIT.BAT végez), olvassuk el a README.DOC-ot!

Lemezszám: M041

Név: Nyákterv

Szerző: Szabó Attila

Leírás: A Pascal forráskódban megírt Graph grafikus nyáktervező segédprogram a konvencionális kivezetőkkel készült alkatrészekből álló áramkörök nyomtatott huzalozási ábrájának gyors és egyszerű elkészítésére szolgál. A program furat-orientált működésű, az egyes alkatrészek körvonalrajzai elsősorban az egyes alkatrészek fizikai elhelyezését segítik. A munkaterületen az alkatrészek (és így a nyomtatási rajzok is) alulról, a nyomtatási oldal felől láthatók. (Ez nem szerencsés, mert a szokásos megoldás az alkatrész oldal felőli tervezés.)
Az alkatrészek körvonalrajzai egy könyvtárban helyezkednek el (GRAPH.LIB). Kissé szegényes a választékuk, mert az integrált áramkörből csak a 16-lábú DIP tokozásúak szerepelnek benne.
Ha ki akarunk lépni a programból, nem kér megerősítést, ezért egy rossz mozdulattal minden addigi munkánkat elveszíthetjük. A program képes két vagy több alkatrész megjelölt pontjait automatikusan összehuzalozni, de bizonyos korlátozásokkal: pl. nem húz át vezetéket az IC két kivezetése között. (S ezt kézi eljárással sem tehetjük meg.)
A Graph grafikus nyáktervező segédprogram ezen változata a következő korlátokkal rendelkezik, amelyeket célszerű a munka megkezdése előtt figyelembe venni:
1. A munkaterület maximális mérete 120 x 120 raszterpontnyi, ami 2,54 mm-es rasztertávolsággal számolva 30,48 x 30,48 cm-es nyáklemez tervezését teszi lehetővé.
2. A maximális rétegszám kettő, tehát legfeljebb kétoldalas nyáklemezt használhatunk.
3. Az egy munkaterületen használható körvonalrajz-féleségek száma maximum 20.
4. Az egy munkaterületen használható alkatrészek száma maximum 60.
5. „AUTO-HUZALOZÁS" üzemmódban a kereső lépések száma maximum 30000.
6. A maximális nagyítás mértéke 8.
Mindent összefoglalva, a program elsősorban azoknak a kispénzű amatőröknek készült, akik azt a fent említett korlátokat figyelembe véve is használni tudják.

Lemezszám: M042

Név: WISE SW #1

Szerző: Moravecz László

Leírás: A Clipper és C rutinok gyűjteményét tartalmazó lemezen a \C alkönyvtárban Borland Turbo C 2.00 nyelven írt rutinok, bővítések találhatók. Ezek jó szinten tesztelt eljárások, azonban csak hobbi szoftverek, bár a profi felhasználás távlatával készültek.
A \CLIPPER alkönyvtárban a Clipper Summer '87 nyelvhez készített eljárások találhatók. Ezek a programrendszerekben éles üzemben is nagy biztonsággal használhatók. Itt van egy Clipper információs kártya szerepét betöltő szövegfájl is.
A \UTILITY alkönyvtár egy-egy problémát megoldó segédprogramjai részint hobbi, részint pro-

ÚJDONSÁG!

ALAPLAP POSTA

CSOMAGKÜLDŐ SZOLGÁLAT

Jelenleg 793-féle szoftverből és 115-féle külföldi szakkönyvből válogathat. A lista megtalálható mostani számunk (92/2) lemezmellékletén.
A megrendelt szoftvert vagy külföldi szakkönyvet postai utánvétellel 2 héten belül házhoz szállítjuk.

MEGRENDELÉS

Megrendelem postai utánvétellel az alábbi termékeket.
A vételárat és a postaköltséget átvételekor kifizetem.

A) SZOFTVEREK:

....

....

....

B) SZAKKÖNYVEK:

....

....

....

Dátum:

(aláírás)

APRÓHIRDETÉSI MEGRENDELŐLAP

Kérem, hogy az Alaplap következő számának Mikrobazár rovatában közöljék az alábbi szövegű apróhirdetést:

....

....

....

....

....

(Maximális terjedelem: 300 betűhely.)

PC Turbo Klub

Ezennel belépek a PC Turbo Klub tagjainak sorába. Az egy évre szóló tagsági díjat befizettem, és mellékelem az igazolószelvény másolatát.
A tagsággal járó Alaplapot és egyéb küldeményeket az alábbi címre kérem:

Név: ....

(Intézmény:) ....

Utca, házszám: ....

Helység: ....

Irányítószám: ....

1992. hó nap

....
(aláírás)

INFORMÁCIÓKÉRÉS

Kérem, hogy az itt általam **BEKARIKÁZOTT KÓDSZÁMÚ** hirdetésekkel kapcsolatban küldjenek részemre bővebb tájékoztatást.

01	02	03	04	05
06	07	08	09	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25
26	27	28	29	30
31	32	33	34	35
36	37	38	39	40
41	42	43	44	45
46	47	48	49	50
51	52	53	54	55
56	57	58	59	60
61	62	63	64	65
66	67	68	69	70
71	72	73	74	75
76	77	78	79	80

ALAPLAP 1992/2 FEBRUÁR

FELADÓ:

A) Egyéni érdeklődő:

Név: ....

Utca, házszám: ....

Helység: ....

Irányítószám: ....

B) Vállalati érdeklődő:

Cégnév: ....

Ügyintéző: ....

Utca, házszám: ....

Helység: ....

Irányítószám: ....

Telefon/Fax: ....

Belföldön díjmentesen feladható

Cédrus Kiadó
Pf. 74

Budapest

1441

FELADÓ

Név: ....

Cím: ....

Helység: ....

Irányítószám: ....

Telefon: ....

- ❑ A hirdetés egyéni és egyedi jellegű, ezért kérem ingyenes közlését.
- ❑ A hirdetés kereskedelmi célú.

Mellékelem a soronként (60 betűhelyenként) 300 forintnak megfelelő összeg átutalását igazoló bizonylat másolatát. (Címzett: Cédrus Kiadó Kft. 1441 Budapest VIII., Reguly Antal u. 8. Bankszámlára: Általános Értékforgalmi Bank 204-19417 számlaszám.)

Belföldön díjmentesen feladható

Cédrus Kiadó
Pf. 74

Budapest

1441

ALAPLAP

Belföldön díjmentesen feladható

Cédrus Kiadó
Pf. 74

Budapest

1441

FELADÓ:

Név: ....

Cég: ....

Utca, házszám: ....

Helység: ....

Irányítószám: ....

Telefon/Fax: ....

Belföldön díjmentesen feladható

Cédrus Kiadó
Pf. 74

Budapest

1441

A LEMEZMELLÉKLET TARTALMA:

- Példaprogramok a Modula 2 sorozathoz
- EGA és VGA karakterletöltő (CRT)
- Videomód-merevítés (Keepmode)
- Gépírási oktatóprogram (demó változat)
- Rajzoktatási szemléltető (demó változat)
- A GEM operációs rendszer — XVIII. rész
- Alaplap Posta (Szoftverek és könyvek jegyzéke)
- A SolarSoft 1991. évi sikerlistája
- Játékok: 1. Égitámadás (Jay-mine) 2. Bűvös karikák (Trisk)



Nekünk a biztonság a fontos.

Mi Polaroid mágneslemezt használunk.

PC SZOFTVEREK

A régi helyen új árakkal, új termékekkel

CLARION PROF. DEV. 2.1	68.000
CLARION DATABASE 3 LEM 2.0	5.000
CLARION DISTRIBUTION KIT 2.0	18.000
CLARION GRAPHICS LEM 4.1	18.000
FOXGRAPH	25.000
FOXPRO 2.0	66.000
FOXPRO 2.0 DISTRIBUTION KIT	42.000
FOXPRO LAN 2.0	99.900
FRAMEWORK IV 1.0	61.000
LAN ASSIST PLUS 3.0	36.000
LAPLINK PRO 4.0	16.000
MATRIX LAYOUT	25.000
MS C 6.0 PDS	38.000
MS MACRO ASSEMBLER 6.0	14.000
MS QUICK C F/W	19.000
MS VISUAL BASIC	19.000
MS WINDOWS DDK	46.000
MS WINDOWS SDK	46.000
MS WORD 5.5 /GRAMMATIK	32.000
MS WORD FOR WINDOWS 2.0	45.000
MS WORKS 2.0	14.000
MS WORKS FOR WINDOWS	19.000
NORTON UTILITIES 6.01	18.000
NORTON UTILITIES 6.01 UPGR.	9.000
QUATTRO PRO 3.0	16.000
SHOWPARTNER F/X	37.000
STATGRAPHICS 5.0	92.000
TOPAZ	12.000
TOPAZ LAN	18.000
VENTURA PUBL. 3.0 GEM	82.000
WORDPERFECT 5.1	39.900
ZORTECH C++ 3.0 DEV.	45.000

Figyelem, áraink a 25%-os áfát nem tartalmazzák.

Cédrus Floppyland Kft.
1056 Budapest, V., Váci utca 84.
Tel./FAX: 1182-651

INFORMÁCIÓKÉRÉS: 24 ▲

KESZO Kft.
1145 Bácskai utca 3/b. 1. em.
Tel./FAX.: 252-91-48

Szoftver kis- és nagykereskedelem raktárról
Vidékre ingyenes házhozszállítás!!!
(Áraink az ÁfÁ-t nem tartalmazzák.)

DOUBLE-UP:
- 1-2-3 FOR WINDOWS 1.0A
- AMI PRO 2.0 FOR WINDOWS
- ADOBE TYPE MANAGER 1.1

30.000,-

BORLAND C++ 3.0	45.000
BRIEF 3.1	23.000
CLARION PROF. DEV. 2.1	68.000
CLARION DATABASE 3 LEM	5.000
CLARION GRAPHICS LEM 4.1	18.000
CLIPPER 5.01	69.900
CODE BASE++	36.000
COHERENT	17.000
CORELDRAW 2.0	52.000
DESQVIEW 385 2.4	19.900
DIS.DOC PROFESSIONAL	28.800
FLIPPER 5.0	38.000
FLOWCHARTING 3	24.000
GOSCRIPT	14.000
GOSCRIPT PLUS	28.000
LOTUS 1-2-3 FOR WINDOWS UPGR.	19.000
MS QUICK C F/W	19.000
MS WINDOWS 3.0	13.500
MS WORD 5.5 & GRAMMATIK	32.000
MS WORD FOR WINDOWS 2.0	45.000
NORTON BACKUP 1.2 WIN/DOS	14.000
NORTON COMMANDER 3.0	13.000
NORTON DESKTOP F/W	16.000
NORTON EDITOR 2.0	11.000
PAGEMAKER 4.0	74.000
PC TOOLS 7.1	17.000
PROCOMM PLUS 2.0	12.000
QA PLUS 5.1 (test program)	16.000
QEMM-386 6.01	10.000
QUATTRO PRO 3.0	15.000
SOUND BLASTER 2.0	18.000
SUPERBASE 4 V1.3 F/W	60.000
STACKER 2.0	14.000
STACKER 2.0 AT/16 bit card	24.000
WRITER'S TOOLKIT (szótár)	19.000

INFORMÁCIÓKÉRÉS: 25 ▲

GALAX
1113 Bp., Bocskai út 54. Tel./FAX.: 166-75-57

3M lemezek, írásvetítők kedvező árakon!

STAR nyomtatók, lapadagolók széles választékban

CANON faxok, printerek, fénymásolók

Írásvetítők	44 800-tól 130 000-ig
EGA, CGA mono	129 900,–
Színes emuláció VGA	294 700,–
Valódi színes SVGA	585 200,–

PC SZOFTVEREK
Microsoft, Borland, LOTUS, NORTON, Central Point...

Áraink az ÁFA-t nem tartalmazzák.

SZERETETTEL VÁRJUK RÉGI ÉS ÚJ TÖRZSVÁSÁRLÓINKAT!

OKTATÁSI INTÉZMÉNYEK SZÁMÁRA KEDVEZMÉNY!!!

INFORMÁCIÓKÉRÉS: 20 ▲

fesszionális igényeket elégítenek ki.
Az egyes szoftveregységek önálló tömörített önkicsomagolós EXE fájlokban találhatók (ZIP). Az egyes szoftverek eltérő dokumentáltságúak, de mindig ott vannak a forrásprogramok is, fordítási környezetükkel együtt, így nem okozhatnak problémát a haladó és profi felhasználóknak sem, továbbá lehetőség nyílik más verziójú Clipper és Turbo C fordítók használatára vagy Microsoft és más C fordítókra történő adaptálásra. Profi felhasználó ugyanis csak olyan eljárásokat építhet rendszerébe, amelyek követése biztosítva van és forráskódjuk hozzáférhető.

Lemezszám: M043

Név: CLDEC87

Szerző: Báró Csaba

Leírás: Clipper '87 Summer decompiler. A kecskeméti Decompiler Stúdió által kifejlesztett Ariadne Clipper '87S decompiler demó változata. Akinek megtetszik a demó-program, a teljes verziót is megveheti a szerzőnél, de azt már nem shareware-áron.
Az Alaplap 1991. áprilisi száma cikket közölt (51. oldal) a program szerzőjétől a Clipper programok visszafejtéséről. Érdemes elolvasni.

Lemezszám: M044

Név: Ray Tracing

Szerző: Rozgonyi Péter

Leírás: Élethű, a fényvisszaverődéseket is kiszámító testmodell, sugárkövetéses ábrázoló mintaprogram. A program (RT.EXE) a sugárkövetéses (ray tracing) eljárással generál perspektívikus képeket gömbökről és síkidomokról, amelyeket egy vagy több fényforrás világít meg. Bár a program a valóságos látvány kialakításában részt vevő optikai hatásoknak csak egy részét modellezi, mégis meglepően élethű képeket állíthatunk elő vele, különösen ha EGA, vagy még inkább, ha VGA grafikus kártyával rendelkező IBM AT 386-tal kompatibilis számítógépen futtatjuk. A fejlettebb grafikus kártyákra nemcsak a nagyobb felbontás, hanem az ábrázolható árnyalatok nagyobb száma miatt is szükség van. Ami a sebességet illeti, a koprocesszor nem sokat javít a sebességen a saját aritmetika miatt. A számítás néhány másodperctől akár órákig is eltarthat, a gép sebességétől és a választott felbontástól függően. A program bármely karakter leütésével megszakítható, de csak a már megkezdett sor végén áll le.

Lemezszám: M045

Név: Univerz

Szerző: Bors Judit, Sáfár István

Leírás: CWI-Ventura, Ventura-CWI oda-vissza konverterek. Segédprogramok a különböző formátumokat és kódkiosztásokat alkalmazó szövegszerkesztők közötti adatcsere lehetővé tételére. A lemezen lévő program csak demó-verzió és csak 2 K-nál kisebb fájlok konvertálására alkalmas. A mellé adott kézikönyv viszont a teljes verzióra vonatkozik, ezért a leírásban szereplő néhány állomány hiányzik az ARJ 2.0-val összepakolt demó-csomagból. A teljes Univerz program viszont nagy fájlok konvertálására is alkalmas.
Futtatásának feltételei:
— AT-kompatibilis számítógép.
— MS-DOS, 3.3-tól felfelé.
— 250 kilobájt szabad memória.
— A merevlemezen 300 K szabad hely.
Sokszor előfordulhat, hogy adatállományainkat más formátumban szeretnénk felhasználni, vagy mások által más formátumban készített állományokat kívánunk hasznosítani. Ilyen esetekben használható jól az Univerz program.

Lemezszám: M046

Név: Penna

Szerző: Soltész András

Leírás: Hazai fejlesztésű, Turbo Pascalban megírt szövegszerkesztő program. Magyar nyelvű menük, magyar helpek. Titkárnőknek, kezdőknek ajánlható.
(Technikai okokból a program kibocsátása valamelyest késik.)

Lemezszám: M047

Név: Labirintus

Szerző: Pintér Gábor

Leírás: Labirintusjáték, pályaszerkesztővel. Részletes ismertetése megtalálható az Alaplap 1991. novemberi számában.

Lemezszám: M048

Név: CHKVIR

Szerző: Leitold Ferenc

Leírás: Leitold Ferenc CHKVIR 5.x víruskereső programjának demó változata. Korlátozása a regisztrált verzióhoz képest, hogy csak detektál (mintegy 900 vírust ismer fel), és vírusirtást nem végez. Alkalmazása elsősorban a magyar vírusátiratok ellen célszerű. A SCAN új (84-es utáni) változatai mellett ez a program is felismeri a Dir2/FAT (Cluster Buster) vírust.

Lemezszám: M049

Név: A PC-Elektronika lemezmelléklete

Szerző: Kónya László

Leírás: A lemez a Műszaki Könyvkiadó Elektronika sorozatában 1991-ben a szerzőtől megjelent PC-Elektronika c. könyv (ISBN 963 10 9011 6) lemezmelléklete. A könyvben számos program és nyomtatott áramköri terv is szerepelt, s azokat az egyszerűbb felhasználás érdekében külön mágneslemezen jelentették meg. A .PCB kiterjesztésű fájlok a Wintek cég SmartWork nyáktervezőjével készültek.

Lemezszám: M050 (HD)

Név: Biblia Demo

Szerző: Biszak Sándor, Arkanum Bt.

Leírás: A Biblia teljes szövegének adatbázisát készítették el az Arcanum Bt. munkatársai

IBM PC gépen használható formában. Eredetileg CD-re készült a program, de átdolgozták 5,25"-os lemezeken terjeszthető formára is.
Az adatmennyiségre jellemző, hogy a regisztrált változatban a csomag a jelenlegi legjobb tömörítővel, az ARJ 2.22-vel is csak 5 db HD (1,2 MB-os) lemezen fér el. A shareware változatba ennek egyik lemezét (Mózes 5. könyvét) dolgozták bele, de az is önállóan használható.

Lemezszám: M051

Név: Autóelszámolás

Szerző: Boros László, Pneumatika Bt.

Leírás: Autóelszámolás Clipperben. A Pneumatika Bt. (Bosch képviselet) támogatásával készült BAUTO.EXE program public domain jelleggel terjeszthető. Semmi korlátozást nem tartalmaz, teljesen funkcionális, magyar nyelvű segédinformációk támogatják a munkát. A fejlesztő neve, címe és telefonszáma megtalálható a programban. A bejegyzett felhasználóknak szükség esetén egyedi igényeiknek megfelelően módosítja a programot.

Lemezszám: M052

Név: GS-Auftrag

Szerző: Kertész Zoltán

Leírás: A Németországban több tízezer példányban bejegyzett, az üzleti levelezést és a számlázást támogató GS programcsalád tagja. A német cím ne ijesszen el senkit, a program 1.61-es verziója teljesen magyarítva kerül a polcra. Akinek megnyeri a tetszését a shareware változat, az jó magyar forintért is megveheti a teljes programot a magyarországi fordítónál és terjesztőnél, Kertész Zoltánnál.
A shareware verzió leírása is magyar nyelvű, ezért kapott helyet a SolarSoft könyvtár magyar szekciójában, bár a program eredetileg német fejlesztésű. (A programcsalád további tagjainak magyarítása is előkészületben van, s azok a magyar viszonyokhoz a lehetőségekhez képest tartalmilag is igazodnak majd.)
A GS-Auftrag kis- és közepes cégeknek megfelelő levelező és számlázó program. A GS-Auftrag a sokkal drágább kereskedelmi programok minőségével is vetekszik. Felhasználási területek:
— Komplett, hálózatos számlázás, vevők, cikkek, raktárkészletek, követelések kezelésével.
— Levelezés a rendeléssel kapcsolatban az ajánlattól a szállítólevélen és a számlán át a jóváírásig.
— Automatikus vagy egyedi fizetési felszólítások.
— A forgalom nyilvántartása vevők és cikkek szerint.
— Címkézés, listázás, kiértékelés, statisztikai kimutatás.
— Formátumok változatos beállításának lehetősége a sok kiegészítő program segítségével.
— Adatátviteli lehetőség a GS-ADRESSEN-be és GS-EAR-ba, címek és követelések átadása és átvétele.
Külön kérésre a csaknem teljes Clipper forráskódot is szállítják.
Teljes változat 14 900,– Ft.
Teljes változat + forráskódok 24 900,– Ft.
Hálózati teljes változat 24 900,– Ft.
Hálózati teljes változat + forráskódok: 34 900,– Ft.

Lemezszám: M053

Név: 1FK

Szerző: Kis Zoltán Gergely, Földi János

Leírás: A Pankotai Állami Gazdaság (Szentes) programozói által készített főkönyvi könyvelési rendszer. A program villámgyorsan dolgozik, magyar nyelvű helpeket ad, és könnyen megtanulható. A shareware változat egy némileg korlátozott demó, aminek kipróbálása után valószínűleg sokan megveszik a teljes verziót is. A bejegyzett felhasználók megkapják a kézikönyvet és a további programfejlesztés során elkészülő új változatokat. A program kézikönyve a Floppyland szakboltban megtekinthető.

Lemezszám: M054

Név: Antivir-2

Szerző: Többek

Leírás: Antivírus shareware-programok. Különböző magyar fejlesztésű víruskereső és vírusirtó programokból készült shareware-összeállítás. Leitold Ferenc CHKVIR programjának csak kereső funkcióval rendelkező demó-verziója külön az #M048 sorszámú SolarSoft lemezre került, itt pedig a használhatónak ítélt többi program szerepel.

Lemezszám: M055

Név: TTL IC katalógus

Szerző: Pete László

Leírás: A program és a katalógus különösen oktatási célokra alkalmas, de mindenkinek ajánlható, aki áramköröket tervez TTL IC-k felhasználásával. A program C-ben készült, adatállománya viszont közönséges ASCII szövegállomány és tetszőleges szövegszerkesztővel bővíthető. A katalógus shareware demó-verziójának adatállománya nem teljes, a teljes verzió a szerzőnél külön 499 forintért (plusz postaköltség) megvásárolható. A programból nyomtatni is lehet.

Lemezszám: M056

Név: Másolásvédelem

Szerző: Csendes Zoltán

Leírás: Kulcslemezes másolásvédelem demó-változata. A Scriptum angol-magyar és magyar-angol szótárának mágneslemezes változatán is ez a kulcslemezes másolásvédelem van. Nem büntető jellegű, csupán nem engedi a védett programot a kalózmásolaton elindítani.
A demó-változat ez esetben annyit jelent, hogy ezzel a programverzióval egyetlen azonosító kód telepíthető az összes lemezre, míg a teljes változatban a lemezazonosító kód lemezről lemezre változik.

Van Önnek XT-je?

Szeretne inkább egy AT-t?

Számítógépét részegységek cseréjével már 16 450,– Ft-tól **átalakítjuk** 12 MHz-es AT-re!

Garanciával!

Ha Ön szereli, akkor csak 13 950,– Ft!

Fordulj hozzánk bizalommal!

Szolinfo Kft.

Tel.: 173-6637
182-2646
166-5413

INFORMÁCIÓKÉRÉS: 27 ▼

Canon «««««««««««««««««««««««««««««« Canon

VÁLTSON SZÍNESRE

Canon

MÁSOLÓGÉPPEL!

CSÚCS AMIT TUD:

- Kicsinyítés
- Nagyítás
- Montírozás
- Tükörkép-készítés
- Képismétlés
- Poszter készítés
- 17 millió színárnyalat



Győződjön meg róla!

fénymásolás

TONER KFT

1095 Budapest, Mester utca 21.
Tel.: 113–1687, 134–3516

Canon »»»»»»»»»»»»»»»»»»»»»»»»»»»»» Canon

INFORMÁCIÓKÉRÉS: 14 ▼

szoftver ABC Kft.

☎ : 201-6891
201-2011/131
: 201-8619
✉ : 1277 Budapest 23. Pf: 45.

Rövid határidővel szállított szoftvereink:

(Ár ÁFA-nélkül!)

DOSHun	6.000	Norton Anti Virus	11.500
Ekszer	45.000	Norton Backup	14.000
Napló 2000	7.900	Norton Backup for Windows	14.000
WinHun	6.000	Norton Commander	13.000
		Norton Desktop for Windows	16.000
		Norton Editor	11.500
Adlib Pers. Music System	19.900	Norton Utilities	15.500
Adobe Type MGR Plus Pak	17.800	Novell Btrieve for Windows	60.500
Adobe TypeManager	10.000	Novell Netware 2.2 5-User	62.500
Aldus Pagemaker	69.000	Novell Netware 2.2 50-User	232.000
Ami Professional	45.500	Novell Netware 3.11 20-User	229.900
Anti Virus +	14.900	Novell Netware 3.11 100-User	459.000
Artline	52.500	Novell NetWare Lite	9.900
Carbon Copy	19.500	Novell XQL	77.500
CC:Mail Fax View	120.000	Novell Xtrieve Plus	44.200
CC:Mail Gateway	142.000	Object Vision	19.000
CC:Mail Import/Export	115.000	On Target	32.500
CC:Mail link to UNIX Mail/uucp	79.900	On Track Disk Manager	9.000
CC:Mail Post Office Pak f. Windows	59.900	OrCad PCB Layout	204.000
CC:Mail Remote	37.500	OrCad VST	176.000
Charisma	42.000	PackRat V for Windows	37.000
Checkit V3.0 /Hardware-Diagnos./	13.900	Paradox	47.500
Chiwriter Professionell	42.000	PC Anywhere IV	16.900
Clarion Profess. Developer	78.000	PC Cosmos	8.900
Clipper 5.01	75.000	PC Paintbrush IV Plus	18.000
Corel Draw 2.0	48.000	PC Tools 7.1	12.000
CP Anti-Virus	9.500	PerForm Pro for Windows	65.000
Crosstalk for Windows	22.000	Personal Rexx	19.000
Dataperfect	34.500	PharLap 386 / VMM	27.500
DBFast / Windows	34.500	PhotoStyler	94.000
DBXL	22.000	PopDrop Plus	10.000
Designer	49.500	Presentation Team	44.900
Deskview 386	22.000	Printer Assist	25.900
Desqview Qemm 386	12.500	Printshop	7.000
Desqview QRam	9.500	Procomm Plus	11.000
Disk Optimizer	7.900	Publishers Paintbrush Windows 3.0	39.500
Draw Perfect	39.900	Publishers Type Foundry	46.500
Draw Plus	13.000	Q & A	36.000
Easyflow	19.000	Q Assist	21.000
F & A	48.800	Quattro Pro	22.000
FaceLift /Bitstream/ 13 Fonts	12.000	Quicksilver	42.000
Facelift for Postscript	20.000	R & R Clipper/Foxbase Modul	7.000
Fontasy	12.000	R & R Rel. Report Writer	22.000
Forest & Trees	49.000	Reflex	24.500
FoxPro	61.490	SCO Unix 3.2 Dev. Pack	97.000
FoxPro LAN	104.000	SCO Unix 3.2 Oper. Sys.	84.000
FoxPro Toolbox	54.400	SCO Foxbase Plus 386	69.000
Framework IV	64.000	SCO TCP/IP Dev. Sys. for Unix 386	26.000
Generic 3D Drafting	32.000	SCO Xenix 386 Oper. Sys.	74.500
Go Script Plus	28.000	Show Partner	11.000
Grammatik IV for Windows	12.500	Show Partner Picture Pack	22.000
Halo Windows Toolkit	57.500	Sideways	13.500
Harvard Graphics	54.500	Sit Back for Windows	14.000
Harvard Project Manager III	57.000	Smalltalk V	12.900
Hijaak	19.900	Smalltalk V Windows	44.000
Intel LANShell	82.000	Smarterm 320	17.500
Intel LANSpool 386	97.000	Software Bridge	13.900
Intel LANSpool for LAN Manager	65.500	Software Carousel	12.000
K-Edit	17.500	Sound Blaster	21.000
LAN Assist Plus	32.000	Source Print	13.500
Landmark Speed Test	5.100	SpeedStor	12.000
Laplink V	14.500	SPF/PC Editor	22.000
Lotus 1-2-3 for Windows	55.000	SPSS/PC+	121.000
Map Assist	33.000	Stacker Harddisk Utility	13.000
MathCad for MS Windows	49.000	Statgraphics	82.000
MathType for Windows	27.500	Superbase IV	64.000
Matrix Layout	24.000	SuperCalc	42.000
MS C Compiler	44.000	SuperProject Expert	75.000
MS DOS 5.0 Update	7.700	Timeline	69.000
MS Excel	43.000	Turbo Pascal for Windows	24.900
MS Flugsimulator Designer	5.500	Ventura Publisher Gold 3.0 WIN	89.900
MS Fortran PDS	43.500	Vitamin C	38.000
MS Macro Assembler PDS	18.000	VM / 386 Multiuser	66.500
MS Pascal	26.000	WinConnect	12.000
MS Project for Windows	62.000	Window Base	51.000
MS Quick C for Windows	16.900	Windows Maker Prof.	92.500
MS Visual Basic	17.500	Windows Word for Word	11.000
MS Windows 3.0	12.000	Winfax Pro	12.900
MS Windows Dev. Kit	38.900	Wingz for Windows	54.900
MS Windows Entertainment Pack	5.500	Wordperfect 5.1	46.900
MS Word 5.5	37.000	Wordperfect Library	18.500
MS Word 5.5 Multispeller	12.000	Wordperfect Office	18.500
MS Word Exchange	7.200	Wordstar 6.0	44.000
MS Word for Windows	43.900	XTree net Advanced	55.900
MS Word for Windows Multispeller	11.700	Zinc Interface Lib. 2.0 Borland	39.000
MS Word for Xenix 386 / Unix 386	94.000	Zortech C++ Developers Ed. V3.0	53.500
MS Works for Windows	22.000	Zortech C++ for Windows V 3.0	35.600
Nantucket Tools II	56.900	Zortech C++ Videokurs 6 x VHS/PAL	39.000
Netroom Single User	9.900	Zortech C++ Views	44.000
NewsMaster II	8.900		

Ami ide nem fért, azt is nálunk keresse!

INFORMÁCIÓKÉRÉS: 12 ▼

Hozzáférés — két akadályon áthágva V.

A rendszer kiteljesítése

Ha már hozzáfogtunk a veszélyek elhárításához, természetesen minél komplettebb megoldásra törekedtünk. Befejezésül most erre szeretnénk rámutatni. Végül pedig utalunk néhány további lehetőségre is.

Mi történik a gép kikapcsolásakor?

Ilyenkor az eredeti jogosultságkódok és a fájlok adatai (név, első cluster sorszáma, eredeti attribútum) még a helyükön vannak, és ha valaki ebben a helyzetben bekapcsolná a gépet, könnyedén használhatná az adott fájlokat. Ezért — rögtön a gép bekapcsolása után — töröljük a jelenleg felesleges kódokat. (Nem tagadjuk, hogy ezt a COMMAND.COM „megfertőzésével", vagyis csekélyke módosításával kívánjuk megoldani.) S ha a ravasz tolvaj floppyról rendszert indítva próbálna bejutni — a bejárati ajtó kikerülésével —, akkor az 1. lépcsőben ismertetett akadályokkal találja szemben magát.

A válságos pillanat

Eddig szó volt arról, hogy hol tároljuk majd a jogosultsági kódokat, de az még nem került terítékre, miként avatkozik be a 2. lépcső a válságos pillanatban. Ehhez részletesen tárgyalnunk kell a DOS-funkciókat tartalmazó 21h sorszámú megszakítást.

Ez a megszakítás végez el minden fájl-, illetve alkönyvtárműveletet. Assembly nyelven programozva úgy aktivizálhatjuk, hogy az AH regiszterbe betöltjük a kívánt funkció kódját, a többi regiszterbe pedig azokat a paramétereket, amelyeket az adott művelet megkíván, majd kiadunk egy INT 21h utasítást. (Magas szintű nyelvek esetén a leírtakkal megegyező tartalmú gépi kódú rutin lesz a fordítás és szerkesztés eredménye.) A legfontosabb funkciók a 21h-s megszakításon belül:

Hexadecimális tevékenység	Funkciókód
0E	Aktuális lemez kijelölése
19	Aktuális lemez lekérdezése
39	Alkönyvtár létrehozása
3A	Alkönyvtár törlése
3B	Alkönyvtárváltás
11FCBs	Első keresés a könyvtárban (FindFirst)
12FCBs	További keresés a könyvtárban (FindNext)
4E	Első keresés a könyvtárban
4F	További keresés a könyvtárban
0FFCBs	Fájlnyitás
10FCBs	Fájlzárás
13FCBs	Fájltörlés
14FCBs	Fájlolvasás (szekvenciális)
15FCBs	Fájlírás (szekvenciális)
16FCBs	Fájl létrehozása
17FCBs	Fájlátnevezés
21FCBs	Fájlolvasás (randomból rekordot)
22FCBs	Fájlírás (randomba rekordot)
27FCBs	Fájlolvasás (randomból blokkot)
28FCBs	Fájlírás (randomba blokkot)
3C	Fájl létrehozása
3D	Fájl megnyitása
3E	Fájlzárás
3F	Fájlolvasás
40	Fájlírás
41	Fájltörlés
43	Fájl attribútumának megváltoztatása
56	Fájl átnevezése
25	Az interrupt vektor egy elemének módosítása
35	Az interrupt vektor egy elemének lekérdezése
4B	Programfuttatás
00	Program befejezése

Az akció helye

Csak akkor tudunk hatékonyan beavatkozni a dolgok számunkra kedvezőtlen folyásába, ha a megfelelő időben a megfelelő helyen vagyunk. Esetünkben a megfelelő hely a 21h-s megszakítás maga, és a megfelelő idő a rendszerindítást követően minden pillanat. Ez azt jelenti, hogy úgy, ahogy az 1. lépcső esetén a 13h-s megszakításnál láttuk, itt a 21h-s megszakítás elé kell láncolnunk a figyelést végző rutinunkat. Itt aztán vígan és feltűnés nélkül dolgozhatunk. Mivel a rutinnak közvetlenül a rendszer elindulása után kell a memóriába kerülnie (és ott is maradnia), ezt a feladatot is a COMMAND.COM módosításával oldhatjuk meg.

De hogyan is módosítható a COMMAND.COM? Mivel COM kiterjesztésű fájlról van szó, amely az első bájtjától végrehajtható (szemben az EXE kiterjesztésűekkel, amelyek egy ún. relokációs táblázattal kezdődnek), egyszerűen beszúrhatunk az elejére egy, összesen 3 bájtos ugróutasítást, amely a fájl végére általunk elhelyezett rutint aktivizálja. (Mellesleg a COMMAND.COM-ot fertőző vírusok ugyanígy működnek.) Feladatának elvégzése után a rutin visszaállítja a bekebeli állapotokat, vagyis a COMMAND.COM eredeti tartalmát 3 bájttal előrébb másolja a memóriában, majd ráadja a vezérlést.

Ezt követően már minden DOS-funkciókérelem a mi rutinunkon fut keresztül. A rutinban ellenőrizzük az AH regiszter tartalmát: ha nem a figyelni valók közé tartozik az adott funkció, akkor egyszerűen az eredeti megszakításra adjuk a vezérlést, ha pedig érdemes a figyelmünkre, akkor elkezdünk szűrni. Beolvassuk a vonatkozó jogosultságot, és ha a kért művelet elvégezhető, akkor az eredeti megszakításra adjuk a vezérlést, ellenkező esetben pedig a 21h-s megszakítástól elvárható hibakódot küldünk vissza az AX regiszterben. A felsoroltak közül a legtöbb funkció esetén elegendő az itt leírt eljárást elvégezni, ám van két olyan, amely plusz tevékenységet igényel: az átnevezés és a fájl-, illetve alkönyvtárlétrehozás.

Átnevezés esetén, ahogy erre már utaltunk, azokban a jogosultsági táblázatokban, ahol az adott fájl szerepel (ide tartozik az átnevezést végző felhasználó táblázata is), adminisztrálni kell a változást. Ha ezt közvetlenül az átnevezés után csinálnánk, akkor feleslegesen pazarolnánk a felhasználó idejét. Ezért a változtatás paramétereit tároljuk az elkövetkezendő bejelentke-

zésekig, és csak akkor módosítunk, amikor egyébként is beolvasnánk egy táblázatot. Kézenfekvő, hogy addig kell a csomagmegőrző szerepét vállalnunk, amíg az összes érintett táblázatot nem módosítottuk.

Lássuk a fájl, illetve alkönyvtár létrehozásának az esetét. Egy ilyen objektumra vonatkozóan minden jogosultságot meghagyunk az adott felhasználónak, de kizárólag neki. Rajta és a főnökön kívül senkinek semmilyen jogosultsága nincs az újonnan létrehozott fájlra vagy alkönyvtárra vonatkozóan. De azért mégsem vagyunk ilyen szőrösszívűek: egy segédprogram segítségével a felhasználónak lehetősége van az objektum más felhasználó(k) számára történő átadására, mégpedig úgy, hogy egy ideiglenes kulcsszó alapján felvehető, postán maradó küldeményként elküldi neki(k). A feladó — anélkül, hogy a saját felhasználói kulcsszavát el kéne árulnia bárkinek is — megsúgja a címzett(ek)nek a küldemény ideiglenes kulcsszavát, ő(k) pedig egy másik segédprogrammal beíratják a saját jogosultsági táblázatukba a szükséges adatokat. Ezután már a többi (erre érdemes) felhasználó is használhatja az újonnan létrehozott objektumot, persze csak olyan jogosultságokkal, amelyeket a feladó engedélyezett.

A 21h-s megszakítás elé láncolt rutin működésének további részletezése már egyenrangú lenne egy megjegyzésekkel tűzdelt Assembly nyelvű listával, ezért ettől eltekintünk.

A „vendég" fogalma a rendszerben

A „vendég" kifejezés az előzőekben már szerepelt, de a fogalom kifejtése akkor még nem volt időszerű. Most a szűrő 2. lépcsőjének majdnem teljes körű ismertetése után már részletesen tárgyalhatjuk.

Ez a fogalom az egyéni kulcsszóval nem rendelkező felhasználó(ka)t takarja. Ő(k) azért nem kap(nak) kulcsszót, mert a legteljesebb mértékig meg kívántuk hagyni a DOS megszokott felszíni működését, annak minden előnyével és hátrányával együtt. A főnöknek óvatosan kell bánnia a vendégjogosultságok megadásával (például célszerű a fájl- vagy alkönyvtár-létrehozási jogot egyetlen könyvtárra korlátozni). Mindezek mellett a vendégnek lehetősége van arra, hogy a ranglétrán eggyel feljebb lépjen, vagyis kulcsszóval rendelkező felhasználó váljon belőle. Ehhez nem kell más, mint egy program segítségével megadnia magának egy kulcsszót. Ezt a kulcsszót a főnök a legközelebbi belépésekor az első figyelmeztető adatok között megtalálja. Ha ekkor jóváhagyja az új felhasználó jogait (ezek kezdetben azonosak a vendég jogaival), az új jogosultsági rendszer bekerül az adminisztrációba a többiekéhez hasonlóan.

A rendszer megengedi, hogy a „tartós" vendég már a jóváhagyást megelőzően is létrehozzon saját fájlokat és alkönyvtárakat úgy, hogy azok mások által ne legyenek hozzáférhetőek, ugyanis ennek a köztes állapotnak ez az egyetlen és igazi előnye. Ha a jövevény jól viseli magát, a főnök bármikor bővítheti jogosultságait. Ha pedig a főnök nem látja jónak az újdonsült jelenlétét, egyszerűen megszünteti annak kulcsszavát minden jogosultságával együtt (sőt, még a fájljait is törölheti).

Vírusfigyelés és plusz szolgáltatások

A szűrő első lépcsőjének leírásánál láthattuk, hogy azon a szinten milyen védelmet kínálunk a vírusokkal szemben. Itt a második lépcsőben arra van lehetőségünk, hogy árgus szemekkel figyeljük az EXE és COM kiterjesztésű fájlok megnyitását, olvasását és írását. Ha a főnök ésszerűen osztotta ki a jogosultságokat (vagyis a lehető legkevesebb személynek adott írási jogot az említett fájlokra vonatkozóan), akkor kicsi a valószínűsége, hogy egy rosszindulatú, csúszó-mászó vírus ilyen módon okozzon kárt a rendszerben.

A fentebb leírt védelmi rendszerhez a következő segédprogramokat kell rendelni ahhoz, hogy komplett legyen a szolgáltatás:

1. Egy, kizárólag a főnök által használható program, amellyel a jogosultságokat és kulcsszavakat kiosztja a felhasználóknak, majd ezek alapján létrehozza a kulcsszavak és jogosultságok táblázatát.

2. Egy olyan program, amellyel a LOGIN-olás végezhető el.

3. Egy program, amely a hagyományos DIR parancshoz hasonlóan listázza a kívánt könyvtár tartalmát, azzal a különbséggel, hogy ez odaírja a fájlok és alkönyvtárak neve mellé a felhasználónak az adott objektumra vonatkozó jogosultságát.

4. Egy olyan programpár, amellyel lehetőség van egy vagy több, saját létrehozású objektum átadására más felhasználó(k)nak. Az egyikkel az objektumok kijelölése, az ezekhez tartozó jogosultságok megadása és az átadási kulcsszó meghatározása (a küldemény feladása), míg a másikkal a kulcsszó megadása után az adott objektumok adatainak a saját jogosultsági táblázatba beíratása (a küldemény átvétele) végezhető el.

5. Szükség van egy olyan programra, amely a „vendég" számára lehetővé teszi a felemelkedést, vagyis hogy saját kulcsszóhoz juthasson.

6. A rendszer működéséhez gyakorlatilag nélkülözhetetlen programokon kívül tervezünk még egy olyan programot is, amellyel a felhasználónak lehetősége lenne saját felhasználói menürendszer kialakítására. (Az egyes menüpontok tetszőlegesen aktivizálhatnának minden, a DOS alatt elindítható fájlt — BAT, COM és EXE kiterjesztésűeket egyaránt.)

További két különleges szolgáltatás is elképzelhető. Az egyik egy azonosító mágneskártya szimulálása floppy segítségével. Ez abból áll, hogy a főnök nem egy egyszerűen begépelhető kulcsszót ad a kiválasztott felhasználóknak, hanem egy floppyt, amely a kulcsszót tartalmazza, természetesen jól elrejtve. Ez valódi mágneskártyával is megoldható lenne, de ez sajnos nem áll módunkban.

A másik szintén egy új korlátozás: a főnök megtilthatja a rendszeren kívül formázott floppy használatát. Itt is egy kódot helyezünk el a lemez kevéssé ismert részén, és lemezművelet esetén ellenőrizzük.

Mindkét szolgáltatáshoz a kapcsolódó feladatokat (azok jellegéből adódóan) a szűrő első lépcsője oldhatja meg.

Műhelymunkánkban az információgyűjtés és a kísérletezés stádiumán már túl vagyunk. Megemlítjük még, hogy a már kész rutinok kisebb részét Turbo-C-ben, nagyobb részét pedig Assembly nyelven (itt egyaránt használva MASM- és TASM- fordítókat) írtuk.

Krokovay Károly—Radványi Tibor

Pro memoria!

Az Alaplap szerkesztősége és Cédrus Kiadó Kft néven önállóvá alakult kiadója ez év elején új irodába, a Nagyvárad tér környékére költözött.
Új címünk: 1441 Budapest VIII., Reguly Antal u. 8.
Telefon és fax: 133-1839

DataFlex adatbázis-kezelő rendszer

Szájhagyomány útján terjedt...

A gomba módjára szaporodó felhasználói klubok sora egy újabb taggal bővült. A NJSZT keretében működő DataFlex Klub nemcsak a felhasználók és a fejlesztők közötti információcsere fóruma, hanem a „nagyvilágban" előforduló, közel 240 000 DataFlex-applikációiról is szeretné közvetlenül tájékoztatni a klubtagokat.

MBase+ vagy DataFlex?

Ismeretes, hogy az adatbázis-kezelés a felhasználói végszoftverek egyik leggyakoribb feladata. A DataFlex olyan relációs hálózati adatbázis-kezelő, amely különösebb marketingtevékenység nélkül csendben terjedt el az országban. Elsősorban nagyobb rendszerek tervezéséhez alkalmazzák. Olyan jelentős termelésirányítási, banki, kereskedelmi rendszerek készültek el DataFlexben, mint például a Paksi Atomerőműben, a Pécsi Nitrogénműveknél, a Vám- és Pénzügyőrségnél vagy a Bakony Füszértnél használatos rendszerek.

MBase+ néven elterjedt a DataFlex 2.2 magyar klónja. A rendszert Turbo Pascal MT+-ban írták, és rövid időn belül a 2.3b C-ben írt változatával cserélte le a gyártó cég. Azonban az MBase+-felhasználók nagy része sajnos nem tudta, hogy DataFlex fejlesztővel dolgozott, holott a két verzió között annyi a különbség, mint a Trabant és a Mercedes között!

A fejlesztőrendszer — amely a Data Access Corporation amerikai cég terméke — több mint 120 operációs rendszer alatt működő 4. generációs, relációs, hálózati adatbázis-kezelő. Egyidejűleg 255 adatbázis adatelemei érhetők el. Ezek között bonyolult kapcsolatrendszer építhető ki. Minden adatbázis a verziószámtól függően 10-15 online indexállományt tartalmazhat, amelyekben 1-16 adatmező szerepelhet. Az indexállományokat csökkenő vagy növekvő sorrendben egyaránt lehet rendezni. A nemzeti karakterek is figyelembe vehetők a rendezésnél.

Az adatmezők ASCII, number, date, binary, text vagy overlap típusúak lehetnek. Az overlap a logikai rekord „byte-to-byte" szeleteit tartalmazza, amelyeket elsősorban indexelési célokra használnak önálló adatmező-hivatkozásként. Az adatbázisok adatszerkezetére jellemző a DBMS (data basis management system), vagyis az adatok tömörítés nélkül, egyszerűen tömörítve vagy teljes mértékben tömörítve tárolhatók.

A DataFlex önálló fejlesztőnyelvvel rendelkezik. A szükséges programokat egy standard ASCII fájlként tárolva akár PC-s környezetben is lefordíthatjuk, de a futtatandó programot más operációs rendszer alá bemásolva módosítás, fordítgatás nélkül is azonnal futtathatjuk. A fejlesztőnyelv 4GL szintű elemekkel tűzdelt. Lehetőség van elemi tevékenységekből újabb nagy bonyolultságú tevékenységsorozatok parancsszintű alkalmazására. A DataFlex legújabb verziója objektumorientált programozású, de megtartja a hagyományos procedurális programozást is, így igény szerint mindkettő keverten alkalmazható. Az adatbázis-kezelő memóriakezelésére a VROOMM-technológia a jellemző. Ez nem véletlen, hiszen C++-ban írták, támogatja az egér kezelését, a redőnyt és a rafinált „ablakosmunkát".

A DataFlex filozófiája

A DataFlex filozófiájára jellemző, hogy a képernyőn minden úgy jelenik meg, ahogy azt egy szövegszerkesztővel megrajzolnánk. Egy-egy konfiguráció ebből maximum 255 darabot tartalmazhat. Ehhez az adatstruktúra definiálása és létrehozása után be kell vinni az adatokat az adatbázisokba. A szükséges módosítások, törlések elvégzése után különböző szempontok szerint gyűjthetjük ki az adatokat. A DataFlex nemcsak DBMS-formátumokból képes adatokat kigyűjteni és a képernyőn szerkesztett kép alapján nyomtatóra, illetve fájlba küldeni, hanem más rendszer részére is generál bemenő-adatokat: Paradox, dBase, Clipper, Lotus 1-2-3, ASCII, DIF, SYLK, WordPerfect, MS Word, WordStar fájlokból készíthet riportot, generálhat SQL forrásprogramot, és adhatja át az adatokat például a Lotus 1-2-3 részére.

Egy egyszerű mintaprogram

Az elmondottakat a következő egyszerű DataFlex program illusztrálhatja. Szeretnénk olyan adatbázist készíteni, amelyben nevek szerepelnek lakcímmel. Cél, hogy a rendszerben szereplő adatok ábécé szerint rendezve, illetve városnevek, utcanevek és irányítószám szerint legyenek lekérdezhetők. Az alábbi képernyőképen megvalósuló komplex procedurális DataFlex program a következő:

A program egyszerre akár 100 terminálon is futtatható online módon az operációs rendszerek széles választékán, eszközfüggetlen. A megfelelő adatmezőn állva a PgUp-PgDn gombokkal az adatelemek fellapozhatók akár részkulcsos kereséssel is. Az adatok megjelennek a képernyőn, módosíthatók, törölhetők. Ha egy nevet begépelünk, az AUTOFIND parancsra automatikus keresést hajt végre. Sikeres keresés esetén minden adatot megjelenít a képernyőn, az EDIT funkcióra váltva minden adat módosítható. Ellenkező esetben új rekordot készít, amelyet az installálásnak megfelelően a már töröltek helyére jegyez be (dinamikus adatbázis-kezelés) avagy a meglevők mögé függeszt. A NEVEK adatbázis IRANYITOSZAM nevű mezőjébe csak az előírás közötti értéket fogadja el, tehát ellenőrzi a bevitelt.

Ha a felhasználó segítséget igényel, a HELP képet kell megjeleníteni, majd visszatérni arra a képre és adatmezőre, amelyből ezt a segítségkérést igényelték.

Ha újabb adatbázist kell készítenünk, és ebben szükségünk van a fenti ada-

tokra, egyszerű hivatkozással az új adatbázist hozzákapcsoljuk a meglevőhöz, és a már eddigiek minden adata rendelkezésünkre áll. (Nem kell egy vállalatnál egy embert minden részlegben felvenni, elég egy helyen, és mindenki ehhez kapcsolódik.) Mindez online módon azonnal aktualizálva, mindenki részére érvényesen jelentkezik.

Client—server architektúra

A DataFlex 3.0-ás verziója nagy kihívást jelent a programozóknak az objektumorientált programozás technikájával. Nem kell a C++ nyelvet ismerni, az objektumorientált programozás lehetőségei mégis teljes mértékben alkalmazhatók. A felhasználó bizonyára szívesen „egerészik" a különböző objektumok és adatok között: módosíthat, javíthat, törölhet. Felhasználhat még olyan adatokat is, amelyeket szkennerrel olvastak be. Ugyanakkor egyszerű kör- és hasábgrafikonok is készíthetők, amelyek mindössze három-négy paranccsorral programozhatók. A grafikai modult egyelőre nem fejlesztették tovább, de nem elképzelhetetlen a látványos előrelépés. Aki viszont szereti a grafikus felületet, az Windows 3.0 alatt is futtathatja rendszerét.

A DataFlex-fejlesztőket munkájukban támogatja a Flexline folyóirat, amely tartalmazza a legújabb fejlesztéseket, és ötleteket ad egy-egy applikációhoz. A francia, a holland, a belga, a svéd és az angol DataFlex-fejlesztők a fejlesztési elképzelésekben új adatbázis-kezelő eszközök megvalósításán dolgoznak: ilyen az SQL-, valamint az adatbázis-server. Ezekben az eszközökben egyértelműen a client—server kapcsolat kiépítése a döntő. A client a szükséges adatrekordokat manipulálja, míg a tényleges adatbázis-karbantartás az erre a célra felkészített serverekben történik. Ezzel szemben a user—server kapcsolatban a useré a döntő szó, majd a manipuláció teljes eredményét visszapumpálja a serverbe.

DataFlex-specifikációk

Maximális DBMS fájlszám 255 db
Maximális mezőszám/fájl 255 db
Maximális index/fájl 15 db
Maximális mező/index 16 db
Maximális indexelemhossz 256 bájt
Maximális fájlhossz OP határolt
Maximális rekordszám/fájl 16,7 millió
Maximális rekordhossz 16 kbájt
Indexmódszer
B+ Multi-level ISAM, Újra definiálható, collate-szekvenciával ellátva
Adatfájl típusa
Pakolt, fix hosszú, véletlen elérésű
Numerikus tárolási forma
Pakolt BCD fix pontos, lebegőpontos
Numerikus precizitás
8 tizedesjegy a tizedesvesszőtől jobbra
Numerikus értékkészlet
Fix +− 99,999,999,999,999.99999999
Lebegő +−1.0e +− 306
Egy időben nyitható adatbázisok száma
memóriahatárnyi (max.255 db.)
Forrásprogramsorok száma
32 000 sor/konfiguráció
Maximális sorhossz 255 bájt
Maximális képablakszám
2,000 (MINIMAX-ban szabályozható)
Maximális kép/program
255 (diszk vagy memóriarezidens)
Maximális indikátorszám
89 (+38 db előre definiálva, Összesen 127 db)
Maximális NUMBER változó 32 000 db
Maximális INTEGER változó 255 db
Integer értékkészlete +−2,147,483,647
Funkcióbillentyű 22, terminálfüggetlen
BREAK-point szint 9 db (opcionális)
Szekveciális fájlkezelés
Sor- vagy vesszőhatárolt
Nem határolt
Eszközfüggő
Parancsfájl felépítése
Védett jel,
Félig fordított
Maximális argumentumhossz 250 karakter
Argumentumtípusok
Text String
NUMERIC fix
NUMERIC lebegő
INTEGER
DATE (extendált Julian)
Parancstípusok
Adatmozgatás és konvertálás
Indikátorok (kondicionálás), Calculate
Nem strukturált vezérlések
Strukturált vezérlések
Képformátumok
Adatmanipulátorok (Entry)
Adatkivitel (Report)
Stringműveletek
Szekvenciális I/O műveletek
Konzol I/O műveletek (terminálfüggetlen)
Makrohivatkozások
Grafikus műveletek
Utility programok
Adatbázis-definiáló
Rendszerinstalláló
Programfejlesztő
Újraindexelő
Programgenerátor
Adatbázis-lekereső
Konfigurálható menü
Compiler
Feltételes indexszekvencia
Definiáló

Alkalmazási területek

Jó szívvel ajánlható a DataFlex minden olyan fejlesztőnek, akinek nagy tömegű adatfeldolgozásra van szüksége. A fejlesztés rugalmasan és hatékonyan megvalósítható az ipar, az államigazgatás, az egészségügy, a mezőgazdaság, a pénzügy, a kereskedelem és a vállalkozási ágazatok területén. A fejlesztést segíti a részletes angol nyelvű dokumentáció. Ugyanakkor magyar nyelven — lásd könyvrovatunkat — többféle kiadvány is segíti a fejlesztők (és a felhasználók) tájékoztatását.

A hálózatban is működő, relációs adatbázis-kezelő használata nem igényel a felhasználótól különösebb számítástechnikai ismereteket. Elég az adatösszefüggéseket és adatstruktúrákat ismerni, és a bőséges utility programok segítségével az egyszerűbb feladatokat maga a felhasználó is megoldhatja, csak bonyolultabb feladatokhoz kell programozói segédlet. Tekintettel arra, hogy a DataFlex a legkisebb PC-től egészen a mainframe-ekig használható, és futtatása — nemcsak a fejlesztők, hanem a felhasználók szerint is! — „fantasztikusan" gyors, így hatékonyabb marketingpolitikával még több hazai alkalmazás elkészítése várható.

Starcz Andor

/kep resident

```
                    Nevek & címek  karbantartása

Név:          ________________________________

Város         ________________________

Utca          ________________________

Irányítószám      ___.

PgUp-PgDn lapozás‖ F6 törlés‖ F9 képernyő tisztítás‖Tab részkulcsos keresés
```

/help resident

```
Tisztelt felhasználó!

Bármelyik adatablakon is áll, a már rögzített adatokat a PgUp-PgDn
billentyűkkel fellapozhatja, módosíthatja. Ha az F6-os billentyűt
nyomja le, az adatok törlődnek az adatbázisból!
```

A kalapos emberek ajánlata

LOGITECH

Logitech termékek teljes áruskáláját kínáljuk Önöknek.

Néhány termék a kínálatunkból :

Logitech Dexxa II egér	1.980 Ft + ÁFA
Logitech Pilot Mouse soros egér	5.230 Ft + ÁFA
Logitech MouseMan soros egér	9.460 Ft + ÁFA
Logitech TrackMan Portable trackball	16.730 Ft + ÁFA
Logitech ScanMan Model 32 kéziszkenner	21.440 Ft + ÁFA
Logitech ScanMan Model 256 kéziszkenner	43.270 Ft + ÁFA
Logitech FotoMan digitális kamera	98.000 Ft + ÁFA

Viszonteladók részére magas dealeri kedvezményeket kínálunk.

XENON Communication Kft. - A Logitech cég hivatalos magyarországi disztribútora

1122 Budapest, Városmajor u. 25/a, II/1. Tel/Fax.: 155-1213

INFORMÁCIÓKÉRÉS: 10 ▲

Profi és amatőr zenészek! Számítógépes zenerajongók!

MIDI hangszervezérlő kártyák PC-hez, hangmodulok, szintetizátorok és szintetizátorkártyák, szoftverek, keverők, sőt a teljes **ROLAND-** és **BOSS-**választék a **TRENDEX Kft.** kínálatában!

Ezek a hangok szólnak a legjobban a játékprogramokon is!!
(Roland MT32, CM32L, LAPC-1)

Mintaboltunk, ahol a **ROLAND-** és **BOSS-**eszközök megvásárolhatók, illetve megrendelhetők:
1117 Budapest XI., Fehérvári út 21.

Roland BOSS

TRENDEX
Kereskedelmi és Forgalmazó Kft.
1124 Budapest, Meredek u. 15. Tel.: 186-8981 Tel./Fax.: 166-5785 Fax: 226-4134

INFORMÁCIÓKÉRÉS: 11 ▲

Káoszelmélet a gyógyításban

Agyunkban káosz van — ez tény. Ugyanakkor a káoszban van bizonyos rend, amit a számítógépek is segítettek feltárni. Most a kutatók számítógépekkel igyekeznek modellezni a szellemi rendetlenséget. A betegeket talán még elektronikus pszichiátriai székbe is lehet majd ültetni.

Az elmebetegségek egyidősek az emberiséggel, kezelésükre azonban csak az utóbbi évtizedekben fejlődött ki hatékony terápia. A pszichiátriában — mint az orvostudomány majdnem minden ágában — a számítógépet a problémák modellezésére használják, s annak segítségével dolgozzák ki az új gyógyító eljárásokat. A pszichoterápiai folyamatok modellezésére is komoly kísérletek történtek, hiszen ez a szakterület szellemi nyomozómunkaként is felfogható, amiben a számítógép közreműködhet.

Kenneth Mark Colby amerikai pszichoanalitikus korán elkezdett programozással foglalkozni, és már 1962-től megjelentek tanulmányai a neurotikus gondolkodás számítógépes szimulációjáról. 1991-ben a depresszió leküzdésére írt számítógépes programja készült el, amely a pszichoterápiának az okok felismerését és tudatosítását alkalmazó eljárásaira támaszkodik (kognitív gyógymód). Abból indult ki, hogy a depressziót a negatív gondolkodásmód idézi elő. Persze az is lehet, hogy megfordítva: a negatív gondolkodásmód játszik szerepet a depresszió kialakulásában, de a lényegen, a kialakult helyzeten már nem sokat változtat az a körülmény, hogy melyik ponton léptünk be az ördögi körbe.

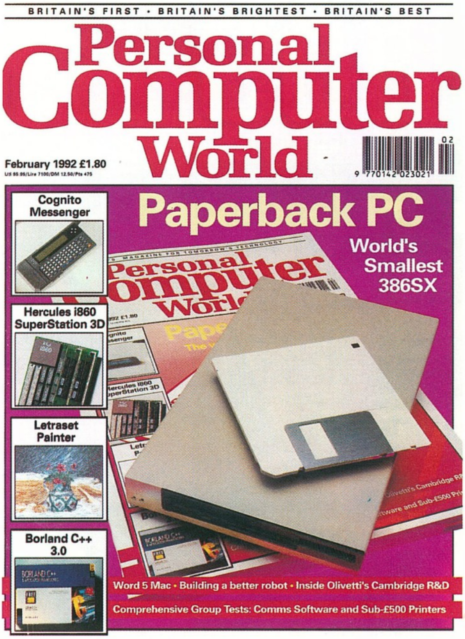
A tudati terápia fő törekvése a személyi számítógépet használva is az, hogy segítsen a depressziós betegek gondolatvilágát megtölteni konstruktív, pozitív tartalmú gondolatokkal. A beteg ugyanúgy természetes nyelvi kommunikációval fordulhat a számítógéphez, mintha az orvosával beszélgetne. Az interaktív dialógus vegyítve van több oktató leckével, és az eredmény elég meggyőzően hat. Az angol nyelv használata még nem tökéletes ugyan, de aki meg tudja bocsátani a néha különös nyelvtani hibákat, annak a rendszer — remélhetően mint terápia is — jól működik.

Sokan úgy gondolják (részben ideológiai okokból), hogy a pszichiátriát, azaz a lélekgyógyászatot nem lehet természettudományos rendszerben művelni, ezért a lélektani kísérleteket és spekulációkat függetleníteni akarják a neurobiológiai alapokon történő agykutatástól. Az idegélettan tudósai közben viszont állandóan újabb összefüggéseket fedeznek fel gondolatvilágunk és agyunk vegyi konyhája, illetve elektromos zsongása között.

Az agy működésének „visszafejtése" igen nehéz feladat, s a számítógépnek fontos szerep jut az idegrendszeri folyamatok szimulációs modellezésében. Különösen a neuronhálózatok elvén alapuló számítógépek fejlesztése termékenyítheti meg a neurobiológia és a számítástechnika együttműködését, s ennek első eredményei már mindkét tudományterületen jelentkeztek.

Egyes kutatók a neuronhálózatos számítógépekkel megpróbáltak modellezni bizonyos agykárosodásokat. Azt például, hogy a homloklebeny sérülése a figyelem torzulásához vezethet, s a beteg nagy figyelemmel tapad olyan tevékenységekhez, amelyek normális körülmények között unalmas, monoton cselekvéssorozatok lennének.

Sem az agy, sem egy PC működése nem érthető meg pusztán annak szerkezete alapján. A személyi számítógép „zsigerei" is ugyanazok Windows, Unix vagy PS/2 alkalmazása esetén. Csak a viselkedés és a funkció vizsgálata teszi a mechanizmust érthetővé. Mivel pedig az agy működési



módja elsősorban elektromos jellegű, azt előbb-utóbbb regisztrálni tudjuk, s akkor közelebb kerülünk a szerkezet és a működés összefüggéseihez.

Az agyműködés elektromos jeleinek jelenlegi regisztrálása (EEG, elektroencefalogram) időrendben közvetíti az adatokat, és új típusú matematikai és számítástechnikai elemzésekhez is kiindulópontul szolgál. Számítógépek nélkül például aligha jöhetett volna létre a modern matematika egyik birodalma, amit nem-lineáris mozgáselméletnek vagy káoszelméletnek nevezünk. Ezzel az új matematikai eszközrendszerrel nagyon sok biológiai jelenség vizsgálható. A tudósok az emberi agy kutatásához is a káoszelmélet matematikai módszereit használták fel. Például nyilvánvalóan megállapítható a káosz az EEG jelekben. A szemünk előtt táncoló görbék túlnyomó része „zajnak" — véletlen „limlomnak" hat. Csak a káosztesztek tudják kimutatni a bennük lévő rejtett struktúrát.

A kaotikus rendszerek szigorú és egyszerű törvényeket követnek, mindazonáltal viselkedésük soha nem jósolható meg előre a nagyon távoli jövőre vonatkozóan. Ez a kiindulási helyzet feltételei iránt megnyilvánuló extrém érzékenység következménye, amit „pillangó effektusnak" is nevezünk.

(Personal Computer World, 1992/február)

Híd a nagyvilágba

A távadatátvitel már régóta nem csak azon sötét alakok eszköze, akik megpróbálnak behatolni titkos adatbankokba, hanem egyre inkább bevonul a mindennapi életbe. Az adatok kényelmes továbbítására több szoftvercsomag közül is válo-



gathatunk. Ezek közül Chris Irwin amerikai programozó D'Bridge nevű programja privát és üzleti célokra egyaránt alkalmas. Az adatforgalmat úgy tudjuk vele lebonyolítani, hogy nem is kell jelen lennünk.

Működésének bemutatására legyen kiválasztott példánk a Fido-Net, amely mintegy 11000 „levelesládájával" a világ legnagyobb hobbihálózata, és főleg az olcsóbb éjszakai tarifával működő automatikus adatátvitelre alapozódik. Aki ebben részt akar venni, annak be kell rendezkednie az érkező „adatcsomagok" fogadására és továbbítására. Az úgynevezett „EMSI-protokoll" révén a hívó számítógép közölni tudja a hívottal, hogy éppen ki akar valamit átküldeni, majd az adatátvitel lebonyolítására az egyik legbiztonságosabb és leggyorsabb rendszert, a ZModemet keresi, de ha ott másik rendszer, például Sealink vagy Xmodem jelentkezik be, a D'Bridge automatikusan átkapcsol arra. A D'Bridge nem fogadja el a küldeményeket automatikusan bárhonnan, hanem az EMSI-ben lévő biztonsági rendszerrel megszűri (igazoltatja!) a hívókat, és ezzel megakadályozza, hogy vírustámadásnak vagy más gonoszságnak essünk áldozatul.

A telefonköltségek csökkentése érdekében a küldemények „becsomagolva", átlagosan mintegy 50 százalékukra tömörítve érkeznek. A D'Bridge mindenekelőtt kicsomagolja és jellegüknek megfelelően — a Fido-Net hálózatban például több száz besorolási kategória van — rögtön szortírozza és automatikusan a „helyükre teszi" őket. Erre a válogatási funkcióra a D'Bridge-nek nem kell külön programot használnia. A legtöbb szoftver a számunkra érdekes, kvázi „előfizethető" témakörök (areas) számát is maximálja (általában 200-ban), a D'Bridge ebből a szempontból is kiemelkedő, mert nem szab korlátot. Igen egyszerű ezeknek a „BBS-témaköröknek" a módosítása is: csupán az Areafix programot kell „értesíteni", ahol a törlendő témakörök megnevezése elé mínusz jelet, az újak elé pedig plusz jelet írunk, és ezzel azonnal át is vezettük a változást.

A szoftver saját terminálprogrammal és szövegszerkesztővel is rendelkezik. Küldeményeinket bármikor megírhatjuk, megkereshetjük, csoportokba rendezhetjük, s ettől teljesen független — általában a kedvező tarifájú éjszakai időpontokra — tudjuk automatikus hívásainkat átterelni. Az időpontot legördülő menürendszeren keresztül írhatjuk be, anélkül, hogy a szövegállományokban kellene keresgélnünk.

A D'Bridge számára a felhívandó állomások kapcsolási számát a Nodelist adja meg, ami a „Fido-Net"-tel kompatibilis rendszerek „telefonkönyve". Ebben definiálhatjuk saját állomáslistánkat is (irodáinkat, barátainkat stb.). Ha faxkártyát használunk, a program azt is közvetlenül kezeli, amely funkció más hasonló programokból szintén hiányzik. A D'Bridge mellett nincs szükség továbbá külön meghajtóra a soros csatlakozóhoz, mert saját maga tud dolgozni a nagysebességű modemekkel, anélkül, hogy tekintettel kellene lennie az MS-DOS 9600 baudos korlátjára.

Ha adatátvitel közben „beszélgetni" szeretnénk a vonal másik végén lévő partnerünkkel, csak be kell kapcsolnunk az Intercom-Chat funkciót, és miközben az adatátvitel zajlik, mi a billentyűzetet használva társaloghatunk a másik számítógép előtt ülővel, vagyis megspórolhatjuk azt a költséget, amibe egy külön telefonbeszélgetés kerülne.

A program az adatátvitellel összefüggő statisztikákat és költségelszámolásokat áttekinthető formában szolgáltatja, egyetlen gombnyomással informálódhatunk a kapott és a kibocsátott üzenetek számáról, költségeiről stb.

A szoftver egyetlen hátránya, hogy kezelésének és kézikönyvének nyelve angol, bár alapfokú ismeretekkel is jól el lehet benne igazodni. Ha a távadatátvitelben még kezdők vagyunk, és a modem üzembe helyezésében sincs tapasztalatunk, nem okoz gondot, mert a D'Bridge automatikusan installálja, az optimális beállítási paramétereket.

(DOS International, 1992/január)

PC-sakk

A sakkjáték számítógépes programozása sokkal egyszerűbb, mintsem azt gondolnánk. A szabályok az elképzelhetetlenül sok lehetséges hadállás ellenére néhány mondatban megfogalmazhatók. A programokban elhelyezett értéktáblázat minden akciófajtát felsorol. Az ellenfél által nem fenyegetett figuránkkal olyan lépést tenni, amely az ellenfél egyik figuráját sem hozza veszélyhelyzetbe, természetesen kisebb értékű, mintha például úgy üthetünk le egy gyalogot, hogy saját bábunkat nem kell feláldoznunk. Az értéktáblázat rugalmasan alakítható. Megfogalmazhatjuk például, hogy az olyan ütésváltás, amelyben mindkét fél azonos figuráit veszíti el (pl. huszárt huszárért), kevesebbet ér akkor, ha mindkét játékos figuráinak száma azonos, és többet, ha az ellenfél már tiszthátrányban van.

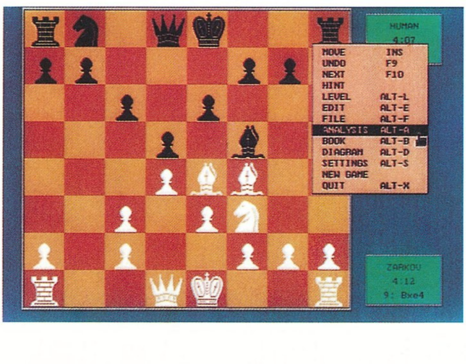
Ha a számítógépnek kell lépnie, mindegyik bábujára kiszámítja az összes lehetséges lépést, és azokat összevetve az értéktáblázattal felállít egy pillanatnyi értéksorrendet. Mivel pedig egy jelentéktelennek látszó lépéssel létrehozott hadállás a későbbiekben igen erős lépések kiindulópontjává válhat, a sakkprogram azt is elemzi, hogy a lépések hogyan befolyásolhatják a későbbi lépésvariációkat, figyelembe véve az ellenfél feltételezhető válaszlépéseit. A számolgatás így a végtelenségig eltarthatna, de az óra ketyeg és lépni kell.

Ilyenkor a számítógép befejezi a hadállás elemzését, és az addig felállított értékrend alapján választja a legjobbnak tekinthető lépést.

A számítógép képes másodpercenként több millió számítási művelet elvégzésére, az ember pedig nem. Hogyan lehetséges akkor, hogy a gép és az ember sakkpárbajában eddig a gépnek nem sok esélye volt. A válasz egyszerű: az ember a hadállásról teljes áttekintéssel rendelkezik, látja, hogy mely bábuk vannak fenyegetett helyzetben, és a táblának éppen mely részei érdektelenek, így figyelmét a kritikus részletekre, a lényegre tudja koncentrálni. Ezzel szemben a számítógép értelmetlen és értéktelen lépésvariációk hatalmas tömegének kiszámítására pazarolja kapacitását. Még a legnagyobb számítási teljesítmény is eltörpül egy sakkjátszmában lehetséges variációk szinte végtelen nagy száma mellett. Nagyot tévednek tehát azok a programozók, akik azt hiszik, hogy programjuk világbajnok lehetne, ha rendelkezésükre állna egy nagyteljesítményű számítógép.

A sakkprogramok egyre jobb eredményeinek forrása csak kisebb mértékben a gyors számítógép, sokkal inkább köszönhető a siker az új programozási koncepciók alkalmazásának. Az első ezek közül, hogy ma már minden jobb sakkprogram rendelkezik úgynevezett megnyitási könyvtárral. Ebben megtalálhatók a nyitólépések viszonylag korlátozott számú jó variánsai. E lépéssorozatok azonban gyorsan lefutnak, s ekkor a modern sakkprogramok egy másik könyvtárt vesznek igénybe, amelyben a korábbi sakkmesterek összes ismert és fontos játszmája megtalálható. A számítógép ebben megkeresi a pillanatnyi hadállás analóg helyzeteit, és kiválasztja a híres elődök legjobb húzásait. Ez az eljárás megengedhető, hiszen a sakkmesterek is hasonlóképpen cselekszenek: fejükben hordozzák a kielemzett régi játszmákat, és a korábban már bevált lépésekhez folyamodnak. A sakkprogramozók azonban most már egyre inkább az álláselemzés új koncepciójával, az optimalizálással foglalkoznak. Azt szeretnék elérni, hogy a sakkprogramok felismerjék a hadállás kritikus pontjait, és hatalmas számítási kapacitásukat arra összpontosítsák. Ha ezt a módszert sikerül tökéletesíteni, akkor az embernek már nem sok esélye lesz, hogy legyőzze elektronikus ellenfelét.

A sakkprogramok játékerejét — akárcsak az emberekét — az Élő-pontszámmal fejezik ki. A világbajnok Kaszparov pontszámát 2800-ra becsülik. A legerősebb sakkprogram jóval elmarad ugyan ettől, de 2500 pontjával igencsak megverné az átlagos egyesületi sakkjátékosokat (1500—1900). Az Élő-pontszám objektivitása a sakkprogramok esetében eléggé vitatott. Nincsenek erőmérő tesztek, az adatok a bajnokságok eredményeinek összehasonlításán alapulnak. A gyártó cégek által megadott érték néha még támpontként is megbízhatatlan. Ráadásul egy sakkprogram tényleges, konkrét játékerőssége közvetlenül függ a számítógép teljesítményétől és a nehézségi foknak álcázott számítási időtől.



Akiket a PC-sakk komolyabban érdekel, azoknak a cikkben ismertetett 4 program bármelyikével érdemes foglalkozniuk (M-Chess v.1.52, Zarkov v.2.5, Rexchess v.2.3, The Chessmachine). Az ötödik, a Battlechess inkább csak kiváló grafikai és animációs megoldásáért érdemel figyelmet — és mert a kezdő sakkozóknak is nyújt sikerélményt.

(DOS International, 1992/január)

A Basic jövője

Sok széppreményű programozási nyelvet láthattunk már jönni — és menni. A PC-k elterjedése óta szilárdan tartja magát a Basic. Vajon megőrzi-e kivívott helyét, vagy eltűnik — tették fel a kérdést a DOS-pódium résztvevőinek. Néhány idézet a válaszokból:

— Tévesek a Basic ellen többnyire önjelölt szakértők által hangoztatott érvek. A „spagetti-kód" hiányos szaktudás esetén majdnem minden programnyelvre ráfogható. A modern Basic compilerek lehetővé teszik a strukturált programozást és nem kevésbé hatékony kódokat írnak, mint a többi nyelv. Ami pedig még tényleg hiányzik vagy lassú, az pótolható az Assembly könyvtárak széles skálájával. Egy feladat különböző részeit mindig az arra legalkalmasabb nyelvvel célszerű megoldani. Például a képernyőkijelzést és az ablaktechnikát Assembly-ben, a komplex stringfeldolgozást Basic-ben, egy adatbank bináris ágait C-ben. Senkinek nem jutna eszébe egy operációs rendszert Basic-ben megírni, de aki ágazati szakterületeken vagy cégen belüli feladatoknál gyorsan és gazdaságosan akar eredményeket elérni, annak a Basic a fő segítőtársa. (Harald Zoschke, Zoschke Data GmbH.)

— A Turbo Basic, a Quick Basic vagy a Power Basic megjelenésével a C és a Pascal komoly konkurenseket kapott, mert ezek a compiler jellegű nyelvek megőrizték a Basic egyszerűségét, könnyű megtanulhatóságát és kezelhetőségét, ugyanakkor átvették a versenytársak alapvető szerkezeti elemeit. A Basic-nek tehát van jövője, bár a megrögzött C és Pascal programozók aligha fognak rá átnyergelni. (Jürgen Hückstädt, szakíró és oktató.)

— A Basic széles körű elterjedtsége egyben hátránya is. Bírálói ugyanis általában csak a korábbi interpreter változatokat ismerik, s nem tudják, hogy az új compilerek lehetővé teszik a strukturált programozást is. Aki logikusan az egyszerűbb Basic-et választotta kezdő nyelvnek, most már haladóként, sőt profiként is kitarthat mellette. A programozók sem akarnak a gép rabszolgáiként dolgozni. Nemcsak a felhasználói, hanem a programozói környezetet is barátságosabbá kell tenni. A Basic a programozást gyötrődés helyett örömteli tevékenységgé tudja tenni. (Josef Kirschbaum, Kirschbaum Software GmbH.)

— A mai Basic compilerek számos előnnyel rendelkeznek akár a Pascallal és a Modulával szemben is, például a futási hibákat (runtime error) tekintve, ami professzionális felhasználási területeken döntő jelentőségű. A Basic programok ezért sokkal biztonságosabban működnek, mint a Turbo Pascal szoftverek. Vagy ellentétben a Basic-kel a C „divatnyelv" programozási hibáinak jelentős részét a compiler nem ismeri fel. (Herwig Feichtinger, Shamrock Software GmbH.)

(DOS International, 1992/február)

PC a műszaki ábrázolás oktatásában

Képernyőn a vetület

Tanári pályám során mindig figyelemmel kísértem
a rajzi szemléltetés fejlődését.
A legkorszerűbb módszert
mihamarabb igyekeztem felhasználni.
Néhány esetben az általam elkészített eszköz
közzététele is megvalósult.
Már azt hittem,
az ilyen irányú tevékenységem véget ért,
amikor — a nyugdíjkorhatár közelében —
jött a számítógép,
amelynek hatása egy csapásra
megváltoztatta álláspontomat.
Mi keltette ezt a nagy érdeklődést bennem
a számítógép iránt?
Talán az újdonság —
avagy az ismeretlen vár bevétele?...
Nem!

Régi módszerek megifjodva

Ma már világos előttem, hogy azok a módszerek, amelyek az előzőekben elterjedtek, mind integrálhatók a számítógépbe. Nem is teszek mást, amikor programot készítek, mint végiggondolom, milyen módon is szemléltettem korábban táblán, írásvetítőn, applikációs eszközzel stb. az adott témát, s máris megtalálom az utat a számítógépre való feldolgozáshoz.

A diavetítés megfelelője az egymástól független képek tárolása, s azok tetszőleges sorrendben való bemutatása. Ha egy ábrasor bizonyos változások folyamatát mutatja be, amelyet akárhányszor megismételhetünk, akkor a hurokfilm helyettesítéséről beszélhetünk.

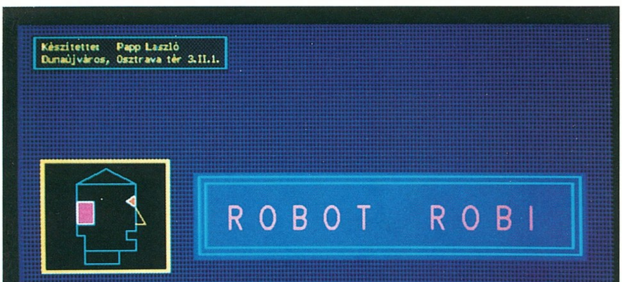
A kevés eltéréssel egymást követő képek mozgófilmet pótolnak, azzal az óriási különbséggel, hogy a folyamat tetszőleges sebességgel játszható le, bármikor megszakítható, s visszafelé is működtethető. Játékprogramjaimban az applikációs módszernél alkalmazott ide-oda rakosgatást, illetve a képek tetszőleges helyzetbe való forgatását szimulálja a számítógép. Nem kell tehát falitábla, dia, applikációs eszköz, film sem a tanárnak (sem otthon a diáknak) az egyes témák megértéséhez, elmélyítéséhez, ha rendelkezésre áll a számítógép és a megfelelő szoftver.

Amikor az árnyék felkúszik...

Közismert tény, hogy egy rajzórán a tanárnak nincs ideje az egyes témák szerkesztését más-más adatokkal elvégezni. Kénytelen egy viszonylag általános példát bemutatni, s legfeljebb szavakkal elmondhatja, hogy mi is történt volna, ha valamelyik érték megváltoznék.

A táblán fáradságos munkával (fehér vagy színes krétával) készített rajz egy villanás alatt megjelenik — például a Perspektíva c. programban — a képernyőn. Természetesen a szerkesztés menetét követő, lépésekre bontott változat is megtalálható a program kínálatában. A leglényegesebb különbség a táblai munkához képest az, hogy bizonyos adatok módosításával (horizontemelés, -süllyesztés stb.) a program újból lefut. Így végignézhető minden lényeges, a törvényszerűség megállapítására alkalmas változtatás, amelyeknek szerkesztéssel való bemutatása teljesen kizárt. Ilyen módszerrel nagyon sok programom működik, de igen jól bevált az „Árnyékszerkesztés"-nél. Itt a képsík-tengely, valamint a fénysugarak helyzete (f1, f2) változtatható tág határok között, aminek következtében létrehozható tisztán az első, illetve a második képsíkra eső árnyék, valamint minden közbülső helyzet. Igen látványos, amikor az árnyékkép felkúszik a K2 síkra.

Az írásvetítő egyik legnagyobb előnyének azt tartják, hogy a fóliákat egymásra helyezve fel lehet építeni egy több fázisból álló folyamatot. A ráépítések számát azonban behatárolja az átvilágíthatóság. A számítógépen az egymásra épülő képek számát semmi



sem korlátozza. Így például a „Hatoldalú hasáb ferde síkmetszése" c. programban a valódi kép leforgatással történő szerkesztésénél jól követhető, hogyan változik a torz hatszög előbb egyenes vonallá, majd egyre jobban nyúlt, végül a valódi képet mutató hatszöggé. A „Hiperbola" nevű programban a kúpot a középvonallal párhuzamosan metsző síkok — a tudás megerősítésén túl — esztétikai élményt is nyújtanak a vetületbe rajzolt hiperbolák sorozatával.

A vetületi képek keletkezésének magyarázatánál szívesen alkalmazott módszer, hogy a tanár a kezében tartott modellt úgy forgatja, hogy abból egyszer felülnézet, majd elölnézet, végül balnézet legyen. A modellt minden esetben odatartja az előre megrajzolt vetület fölé. Az „Átfordulás" c. programban a modellt egy színes axonometrikus kép jelenti, amely elindul a felülnézet felé, s mire odaér, felülnézetté is változik. Természetesen a másik két nézetet ugyanilyen átúszással hozza létre, sőt, egyik vetületből a másik vetületbe való átfordulás is előállítható vele.

Egy kép többet mond, mint ezer szó

A Mikromagazin 1988/4. számában ezt a címet adta Zsadányi Pál annak a cikkének, amelyben az én (TVC-re írt Ábrázolás c.) munkámról írt elemző értékelést. Ennél találóbban én sem tudnám kifejezni azt, hogy bármit is írhatok, a számítógép csodálatos előnyeiről, a látványt, amit nyújt, bármennyi szóval sem lehet visszaadni.

A mágneslemez mellékletre hely hiányában csak a Robot Robi elnevezésű demóprogramrészlet kerülhetett rá, a rövid bemutatás nem is adhatja vissza a gazdag választási lehetőséget, de betekintésre talán mégis elegendő.

Papp László

Program kezdő harkályoknak

A gépírás tanulása mechanikus, sok ismétlődést tartalmazó feladat. Hagyományos menete jól algoritmizálható folyamat, szinte kiált a számítógép után. Lényeges azonban, hogy a körülmények, amelyeket a program teremt, megfeleljenek az iskolai, illetve a szakmai vizsgán támasztott követelményeknek.

A számítógép, a billentyűzet, a monitoron megjelenő karakterek képe — tudjuk jól — könnyen alakítható, így alkalmassá tehető a gépírás oktatására. Tankönyvből tanítva pedig a feladat hasonlít a majdani vizsgához, ahol egy papírról kell az ismeretlen szöveget megadott idő alatt hibátlanul (igen kevés, maximum hatezrelékes hibával) leírni. A feladat lényege tehát a hibátlanság. A hagyományos írógépes gépírástanításban is a hibák javítása a leglényegesebb. Az oktatószoftver ezt a szerepet veszi át a tanártól.

A hibakezelő rutin nem tökéletes, de nem is lehet az. Amit egy ember, különösen egy tapasztalt szaktanár egyetlen pillantással átlát, azt a számítógép csak hosszas hasonlítgatással, keresgéléssel képes — így-úgy — elvégezni. A problémát nem az egyszerűen leírható hibák okozzák (például egy betűelütés), hanem az a kérdés, hogy mikor azonos egy leírt szó a leírandó szóval. Ha az a feltétel, hogy azoknak betűről betűre meg kell egyezniük, a tanuló semmit nem hibázhat. Ha viszont hibázik, akkor azt leleményesen teszi, és a programozónak nem is lehet akkora fantáziája, hogy minden lehetséges szövegkörnyezetben, minden lehetséges hibát pontosan azonosítani tudjon. Mégis elmondható, hogy a javító rutin teszi hasznossá a programot. Az esetleges bakikat pedig remélhetőleg elnézik nekünk.

Működése közben a rutin megkísérli azonosítani a leírt szavakat, és összevetni a tankönyv szövegével. Szó a programnak az a betűfüzér, amelyet egy szóköz vagy egy sorvégjel zár le. Ha egy szóban három hibánál több van, azt nem tekinti azonosnak az etalon megfelelő szavával, de megkísérli megkeresni a teljes szövegben. Ez a sorugrás. Ha így sem tudja azonosítani, akkor javíthatatlannak minősíti, de a szavak sorába beszámítja, tehát a tankönyvben is lép.

A további szolgáltatások csak mellékesek, bár nem elhanyagolhatóak. Reméljük, hogy lévén gyakorló tanárok által létrehozott program, ha számítástechnikailag talán nem a legtökéletesebb, szemlélete miatt mégis a leghasználhatóbb. A program demóverziója a lemezmellékleten található.

Albu László
Alternatív Közgazdasági Gimnázium

A Mikrobazár rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hirdetéseket közlünk.
A kereskedelmi tevékenységet szolgáló apróhirdetések tarifája gépelt soronként (60 karakterenként) 300 Ft. Kérjük, hogy a hirdetés díját a Cédrus Kiadó Kft.-nek az Általános Értékforgalmi Banknál vezetett 204-19417 számú számlájára utalják át, vagy postautalványon a Cédrus Kiadó Kft. címére (1441 Budapest VIII., Reguly Antal u. 8.) fizessék be, a hátoldalon feltüntetve, hogy apróhirdetés. A befizetést igazoló szelvényt a közlendő hirdetési szöveggel együtt az Alaplap szerkesztőségéhez küldjék el: 1441 Budapest VIII., Reguly Antal u. 8.
A nem kereskedelmi célú egyéni hirdetések közlése INGYENES!

ADOK

Enterprise programok eladók. Válaszboríték ellenében listát küldök. 2000 program, sok kedvezmény, ajándék. Cím: Zemen László, 1104 Budapest X., Kada u. 141. fszt. 9.

ÖTÖD-ÖLŐ játék 300 forintért eladó. Cím: Csupor László, 1539 Budapest, Pf. 720. Tel.: 115-4352.

IBM XT/AT játék- és felhasználói programokból óriási választék. Cím: Szőnyi László, 1161 Budapest XVI., Tavirózsa u. 5. Tel.: 184-8471.

Turbo Pascal, Turbo C hívők! Használják ki a VGA és Super VGA kártyák 256 színű grafikáját! VGA/MCGA és Tseng Labs ET3000/ET4000 processzorú SVGA kártyákhoz BGI-k kapkatók! VGA/MCGA BGI: 200 Ft, Tseng SVGA BGI: 600 Ft. Válaszboríték ellenében részletes leírást küldök. Cím: Melis László, 2100 Gödöllő, Mosolygó Antal krt. 24. III. 9.

IBM XT/AT játék- és felhasználói programokat adok-veszek-cserélek. Listát kérek és küldök mindenkinek. Ugyanitt eladó CGA monitorkártya, 360 kB-os floppy + vezérlő, valamint ZX Spectrum 48K + magnó + nyomtató + Centronick interface. Cím: Zalavári Miklós, 9023 Győr, Ipar út 100.

Ingyen juthat a legkülönfélébb **PC-s felhasználói és játékprogramokhoz** az ország egyik legnagyobb programbankjából. Válaszboríték ellenében bővebb tájékoztatót és listát küldünk. Cím: SMID-SOFT, 3672 Borsodnádasd, Népköztársaság út 84.

Programokra, segítségre, cseretársakra van szükséged? Az ASIS megoldja problémáidat! Bárhol laksz, bármilyen géped van, írj! Kérésre ingyenes tájékoztatót küldünk. Cím: ASIS, 1425 Budapest Pf. 729, Tel.: 142-8075.

Ahol az orvos már PC-zik

Kór-szerűen...

A Dél-Pesti Kórház sebészeti osztálya egyik fiatal orvosához kollégái többsége egyvalamiben ma még sajnos nem hasonlít. Ő ugyanis nemcsak szereti és tudja kezelni a számítógépet, hanem mindent meg is tesz, hogy munkája során egyre újabb területeken alkalmazza a PC-t. Megszerezte az informatikus üzemmérnök másoddiplomát is.

Amikor Nagy Péter a Semmelweis Orvostudományi Egyetemre járt (1976 és 1982 között), még nem volt alkalma, hogy gépközelbe kerüljön. Volt ellenben egy fakultatív tantárgy — a diákok mintegy tizedének részvételével —, ahol Texas programozható kalkulátorok segítségével próbálták a számítástechnikai gondolkodásmódnak helyet szorítani a legelszántabb medikák és medikusok fejében. Nagy Péternek az első élményt mégsem ez, hanem egy tudományos diákköri munka jelentette: egy mások által írt Fortran programmal dolgozták fel az egyik klinika hipertóniás betegeinek adatait.

A számítástechnika iránti érdeklődést a munkahely kétségtelen varázsa sem irtotta ki a kezdő dokiból. Az éjszakai ügyeletek unalmasabb óráiban élvezettel vetette rá magát egyik kollégájának öccse által a kórházba bevitt — és még mindenféle háttértárolót nélkülöző — ős-hobbigépre.

Az idők szavára (1985 körül) a kórház vezetése is fölfigyelt, vásárolva az akkori divatnak megfelelően néhány Commodore 64-est a számítástechnikai osztályra. Ekkor ismerkedett meg Péter a Basickel. Azonban sem ezek a mikrogépek, sem a később beszerzett betegfelvevő, dokumentáló — ún. osztályos — rendszerek (amelyek Forth nyelven íródtak, és valamilyen „kőkorszaki" minigépen futottak) nem forgatták föl túlságosan a kórház életét. Az áttörést — először a hardver, majd később a szoftver oldaláról — az 1987-es év hozta meg.

„Tulajdonképpen az én kedvemért került az első IBM-kompatibilis PC az osztályra. Kértem, hogy tovább tanulhassak. A műegyetemre szerettem volna menni, de csak annyi pénzt sikerült szerezni, hogy a Kandó posztgraduális informatikai szakát végezhettem el" — emlékszik vissza a kezdetekre.

A Kandóban információelméletről, kódolásról, a hardver alapjairól hallgatott órákat, és persze dióhéjban megismerkedett a számítógép felépítésével, fejlődésének történetével, a rendszerszervezés alapjaival is. A konkrét, „gépközeli" tananyagban pedig találkozhatott a PC-kkel és a Pascal nyelvvel.

Akinek szinte harcolnia kell egy iskolába való bekerülésért, az általában komolyabban veszi az oktatást. Péter is a túl laza számonkérést panaszolta föl: „Az előadók sokszor nem vették elég komolyan a tanítást, sem a maguk, sem a diákok oldaláról". Ezenkívül több konkrét példát szeretett volna látni, amelyek bemutatták volna a többé-kevésbé laikus hallgatóságnak, hogy mi az, amire lehet és érdemes használni egy PC-t, s mire nem. (A KKVMF különböző szakain végzetteken kívül nemcsak orvos, hanem például kertészmérnök is járt az „osztályba".)

Ma már a doktor úr rendszeresen használja az otthoni vagy a sebészeti osztályon lévő két AT valamelyikét, leggyakrabban dokumentálásra (betegfelvétel, műtéti leírás, levelezés) és statisztikai számításokra (műtéti statisztika). Tervekben sincs hiány: „Jó lenne a gép grafikai képességeit is kihasználni, akár a ma divatos multimédia irányába lépve. Videófelvételek felhasználásával és manipulálásával például sokkal könnyebben és látványosabban lehetne a műtétek egyes fogásait oktatni a medikusoknak. Szintén okos dolog lenne — amennyiben sikerül jobban megoldani a gépi fordítást — az idegen nyelvű cikkek magyarított változatainak tárolása a PC-n. Azt is meg lehetne tenni, hogy az Orvosi Hetilap cikkeit mágneslemezen is hozzáférhetővé tegyék. Mindezek persze újabb beruházásokat is igényelnének, VGA-monitorokat, szkennert, optikai karakterfelismerő programot."

Kíváncsi voltam, hogy orvos kollégái — és általában az egészségügyben dolgozók — hogyan vélekednek a számítástechnika alkalmazásáról. A válasz konkrét példája: az osztály mintegy húsz orvosa közül körülbelül öten használják a PC-t, főleg szövegszerkesztésre. Közülük ketten programot is írnak. A többiek feleslegesnek, esetleg kimondottan zavaró körülménynek érzik a számítógépet, amelyhez nekik kellene alkalmazkodniuk, és ezt nem szívesen teszik. Valóban akad még sok javítanivaló a programok használhatóságán, orvosközeliségén. De az orvosi hivatás jelenlegi szemléletmódja talán még inkább oka az idegenkedésnek. Mintha az orvosokban valami misztikus, már-már sámánisztikus kép élne saját szakmájukról, és sokan ezt a betegekkel kialakítandó viszonyban is szeretnék megőrizni, s a gépben az orvos és a beteg közé betolakodó fölösleges harmadikat látják, amely profanizálja a szándékuk szerint transzcendens viszonyt.

A gép — akárcsak más területeken — természetesen nem pótolja a komplex emberi gondolkodást, az intuíciót, az érzelmi ráhatást és az orvosi munka ezernyi személyes összetevőjét, de rendkívül sokat tud ehhez hozzátenni saját erényeivel: nagy sebességével és konok precízségével. A diagnosztikában a legvalószínűtlenebb eshetőségről sem feledkezik meg, nem hagyja, hogy az orvos bizonyos jelenségek fölött átugorjon. Ráadásul a számítógép tudásbázisából folytonosan lehet tanulni, új öszefüggéseket megismerni. Nem elhanyagolható az általa nyújtott biztonság sem: a bonyolult gyógyszeradagolási számításokat is megbízhatóan és igen pontosan elvégzi. **Tevan Imre**

Hogyan mondjam el neked...

Friss diplomásként tavaly kezdtem tanítani, fizikát egy egészségügyi szakközépiskolában. S mert a hálátlan feladatokból nem volt elég, rögtön elvállaltam az ügyvitel-számítástechnika tantárgyat is. Ha akkor tudtam volna...

Az első nehézség rögtön az indulásnál az iskola „számítógépparkja" volt. Kettő darab HTZ, öt darab Plus/4-es (magnóval és Junoszty televízióval), egyetlen C64-es (floppyval, nyomtatóval). Ennyi. Ekkor még nem jöttem zavarba, a HTZ-ket gyorsan elsüllyesztettem a fizikaszertár mélyére, kipakoltam a Plus/4-eseket egy terembe, és elkezdtem gyűjteni PC-ket — fényképen. Tapasztalatom szerint ugyanis a gyerekek nem képesek a külalak tekintetében absztrahálni: ha a bekapcsológomb nem jobboldalt lent található, hanem hátul fent van, már az üzembe helyezésig sem jutunk el. Elmesélem, hogy a sokféleség oka: az egységekből minden cég azt épít be egy dobozba, amit jónak tart, de minden gép rendelkezhet ugyanazokkal a perifériákkal.

Amikor az anyagi lehetőségekről érdeklődtem, azt a választ kaptam: „Minden van, csak pénzt ne kérj! (De ugye az órarendkészítő programot megírod...)" Szerencsére akadt egy vállalkozás, amely a fejébe vette, hogy támogat minket. Felajánlottak százhúszezer forintot, azzal a feltétellel, hogy azt náluk költjük el hardverre. Így most van egy PC-AT is. Sajnos azonban a programok is pénzbe kerülnek... Ezért jót nevettem azon a minisztériumi felmérésen, amely a vásárolt programjainkat firtatta. Hamar túlestem a kitöltésén: az van, amit valahol lemásoltak nekem, semmi több. Annál is inkább, mert a kisgépeket mire lehet használni így üresen — a Basic tanítására. Erre pedig már elvből sem voltam hajlandó. (Szemléletemet tükrözi az „Intelmek..." c. cikk az Alaplap 1991. decemberi számában.) Oktató — vagy az oktatásban használható egyéb — programért ma Magyarországon pénzt kérni elég aljas tett!

A tárgyi nehézségeken nem túljutva, csak túltéve magam, szembetalálkoztam a lányok masszív ellenállásával. Az első órák mindig a győzködéssel teltek el. Amikor feltettem a kérdést, hogy mire is jó ez a szerkentyű, mély csend volt a válasz. Ekkor cselhez folyamodtam, és azt mondtam, hogy aki a következő órára összeír három olyan alkalmazást, amilyent a többiek nem, kap egy ötöst. Ez bejött: lelkesen gyűjtögettek, és az érdeklődés is nagyobb lett.

A kórházi gyakorlatokon pedig saját maguk győződhettek meg arról, hogy a számítógép nem is az, aminek hívják. Lassan oldódott a görcs is, hogy ez a masina csak a fiúknak való. S akkor lett igazán teljes a sikerélmény, amikor végre valamilyen saját alkotás is megszületett a gépen!

A billentyűzethez szoktatásra kiválóan alkalmasak a Plus/4-es grafikus karakterei. („Rajzoljatok kisautót, vitorláshajót, pálcikaembert stb.") Szinte csodájára jártak az első szövegnek, amely szövegszerkesztővel megalkotva került a faliújságra. Talán nem vagyok Olcsó János, ha azt állítom, hogy ezen a szinten a Botticelli tökéletes rajzolóprogram. S ha nem az, akkor sincs más...

Tudnak segíteni? Akár anyagiakkal, akár jó tanáccsal — mindkettőt szívesen látjuk! Ne feledjék: ezek a gyerekek fognak az orvost, a gazdasági döntéshozót, az autógyári minőségellenőrt... és mindazokat kiszolgáló számítógépek mellett ülni, akikre ön az életben nap mint nap és egyre jobban rászorul.

Zoltai Péter

Legyen-e számítástechnikai „Rigó utca"?

Kicsit karikírozva: Elküldené-e ön házastársát akárcsak egy egyszerű vakbélműtétre is olyan orvoshoz, aki nem tanult 6 évet az egyetemen, nem vett részt igazoltan szakorvosi továbbképzésen, csupán időnként bement letenni a vizsgákat? — Ugye, nem?

Az a helyzet, hogy az informatikai szaktudás nem pusztán technikai fogások halmaza, hanem bizonyos magatartási, problémamegközelítési mód és modell is, amelyet a kollégákkal, tanárokkal való tartós együttlét során lehet és kell elsajátítani. A szakképesítést igazoló oklevél nemcsak azt jelenti, hogy tulajdonosa meghatározott szakismeretnek a birtokában van, hanem azt is, hogy őt a szakma — személyiségét is megismerve — befogadta. Aki a jövőben hozzá fordul, biztos lehet abban, hogy problémáit a szakma általános és specifikus szabályai szerint fogja kezelni. Ha mindezt elvárjuk, akkor a megmérettetés nem történhet csupán néhány órás állami vizsgával. A szakképzés több féléves ideje alatt a hallgató együtt él, együtt dolgozik tanáraival, hallgatótársaival, csoportosan feladatokat old meg. Bebizonyíthatja, hogy képes együttműködni másokkal.

E cikk szerzője rendszeresen részt vesz az állami nyelvvizsgabizottságok munkájában, és biztonállíthatja, hogy a nyelvtudást vagy nyelvtudatlanságot az ottani procedúrák során meg lehet állapítani. (Az más kérdés, hogyan állítjuk be a különböző követelményszinteket.) Mint villamosmérnök, mint a Számalk oktatási üzletágáért is felelős igazgatója viszont ugyanilyen biztonsággal állítja: ésszerű időkorlátok között tartandó állami vizsgáztatáskor nem lehet jó lelkiismerettel eldönteni valakiről, hogy informatikai szakember-e vagy sem. (A szélső eseteket, az abszolút tudatlanokat persze vizsgáztatással is ki lehet szűrni.)

Válaszom tehát a címben feltett kérdésre az, hogy ne legyen. Csupán vizsgák alapján ne adjunk informatikai szakképesítéseket.

Kovács Ervin

Áramkörgyártás személyi számítógéppel

Nagyágyú az íróasztalon?

Kevesen büszkélkedhetnek azzal, hogy valaha is közelről látták egy berendezésorientált áramkörtervező és -gyártó rendszert. Pedig manapság már Magyarországon is gyakran találkozhatunk egy kisebb íróasztalon is elférő BOÁK-laboratóriummal, melynek teljesítménye egyáltalán nem lebecsülendő.

Napjainkban mind több vállalkozás célozza meg a berendezésorientált áramkörök új generációinak kifejlesztését, az alkalmazástechnológia gyökeres átalakítását. Az amerikai Actel és Altera cégek mellett az irányzat egyik elindítója és legjelentősebb képviselője a Xilinx. Azzal a digitális alkatrésszel is a Xilinx cég rukkolt ki, amelyről a tervezőmérnökök már évek óta álmodoztak. Az áramkörrel, amelyet felhasználó által programozható kaputömbnek (Field Programmable Gate Array, FPGA) neveznek, a legkihívóbb feladatok is megoldhatók, mivel a nagy sebesség és integráltság követelménye mellett a piacra dobás ideje és kockázata is csökkenthető. Azokkal az előnyökkel, amelyeket a Xilinx ajánl, a cég a félvezetőipar leggyorsabban növekvő szegmensén, a CMOS technológiájú, programozható digitális eszközök területén lépett a dobogó tetejére. A Xilinx áramkörei ipari szabvánnyá váltak. Angol márkanevük Logic Cell Array (LCA), vagyis logikai cellatömb.

Az LCA áramkörök tokozott kapuáramkörök. Teljes egészében a felhasználó által programozhatók, a konfigurációs információ az LCA áramkörrel egybeépített statikus memóriában tárolható, ellentétben a hagyományos kaputömb-áramkörök . maszkprogramozott technológiájával. A hatékony fejlesztőrendszer alkalmazásával — amelyet szintén a Xilinx szállít — a tervezés, szimuláció, gyártás és tesztelés átlagos ideje akár 10-15 munkanapra is csökkenhet.

Az 1. generáció, a Xilinx XC2000 sorozat 1985-ben került a piacra, és ez a család a kisbonyolultságú programozható logikai eszközök (Programmable Logic Device, PLD) első változatát jelentette. Az XC3000 sorozat, amelynek tagjai mintegy ötszörös bonyolultságúak, 1987-ben jelent meg — erősen megnövelt elemválasztékkal. E 2. generáció megduplázta a rendszer sebességét, megháromszorozta az alkatrészsűrűséget, és a gyors piacra dobás lehetősége továbbra is fennáll. Így a tervezők a technológia fő csapásirányába kerülnek, mivel itt a csúcsot az alkalmazásspecifikus integrált áramkörök (ASIC) széles skálájú használata jelenti. A Xilinx cellatömbök éppen ezen áramkörök kiváltására készültek.

Az LCA áramkörökre alapozó tervezési, kivitelezési folyamat három fő részre bontható: tervbevitel, megvalósítás és ellenőrzés. A Xilinx fejlesztőrendszerrel a tervezés igen olcsó lett, a mérnök a számítógépével — amely akár kaputömbgyártó „laboratóriumnak" is nevezhető — igen hatékonyan dolgozhat. Az áramkör elkészítésének ideje is jócskán lerövidült, a hagyományossal (több héttel) szemben így percekre. Hibás tervezés esetén nincs költséges többletkiadás: az LCA áramkörök a javítás után akárhányszor újraprogramozhatók. Továbbá az ellenőrzés is igen gyorssá vált. A Xilinx fejlesztőrendszer mellett az egyszeri műszaki kiadások (NRE) szinte teljesen elmaradnak, így az LCA kis sorozatban (néhány db/év) is használható, de gazdaságosságát egészen a 10-20 ezer darab/év tételszámú sorozatgyártásig megőrzi.

Az áramkörök felépítése

Az LCA áramkörök kis teljesítményigényű CMOS technológiával készülnek, különböző sebességű és tokozású változatokban kaphatók. A konfigurációs információt az LCA belsejében a statikus RAM tárolja, ami egyszerűbbé teszi a berendezések fejlesztőinek feladatát is. Az alapvető építőelemek a logikai blokkok (ki/bemeneti vagy IOB, és konfigurálható logikai vagy CLB), valamint az azokat összekapcsoló hálózat.

Az IOB-k száma egyes típusokban száz felett is lehet. Konfigurálás során külön-külön programozható a ki- és bemeneti üzemmódok jellege, a belső regiszter használatának módja és a többi paraméter. Az IOB-k kimeneti, bemeneti, kétirányú, valamint nagyimpedanciás üzemmódban működhetnek.

A CLB-k három fő részből állnak: kombinációs blokkból, tárolókból és az összekapcsoló szelektorokból. A kombinációs blokk tetszőlegesen konfigurálható logikai függvénygenerátor. A CLB tárolója két D flip-flop, invertálható órajelbemenetekkel és aszinkron törlőbemenetekkel. Kimeneteik megjelenhetnek a CLB kimenetein, és visszacsatolhatók a kombinációs blokk bemenetére.

Az összekötő hálózat három komponensből áll: általános és direkt összeköttetésekből, valamint a hosszú vonalakból. Az általános célúak rövid vízszintes és függőleges vonalakból álló szegmensek a CLB-k és IOB-k alkotta mátrix elemei között. A szegmensekre a blokkok ki- és bemenetei kapcsolódnak, a találkozásokban állnak a konfigurálható kapcsolómátrixok. A direkt összeköttetések gyors kapcsolatot létesíthetnek a szomszédos blokkok között. A hosszú vonalak — függőleges és vízszintes, kis késleltetésű vezetékek — teljes hosszukban átszelik az áramkört.

Az áramkörben órajelhálózatot is kialakítottak. Ennek gyors vonalai fésűszerűen behálózzák a CLB mátrixot, biztosítva az órajel csúszásmentes eljuttatását minden blokkhoz.

Az LCA áramkör konfigurálásán a bitmintának az áramkör statikus memóriájába juttatását értjük. A memória a külső zavarokra érzéketlen, azonban a tápfeszültség kimaradása esetén információtartalmát elveszti. Gondoskodni kell tehát arról, hogy az áramkör megőrizze vagy minden bekapcsoláskor felvegye a konfigurációját. Amikor telep vagy akkumulátor használata nem indokolt, a rendszert minden egyes LCA

áramkörét bekapcsolás után konfigurálni kell. Önálló rendszerekben az LCA konfigurációs információi külső PROM memóriában tárolhatók. Mikroprocesszoros környezetben a mikroprocesszor buszára perifériaként illeszthetők. Ilyenkor a konfigurálást a mikroprocesszor végezheti kiviteli utasítások sorozatával. Az ilyen konfigurálás magában rejti az áramkör üzem közbeni átprogramozásának lehetőségét is, amely készülékek fejlesztése, tesztelése során is jó szolgálatot tehet.

A fejlesztőrendszer

Kizárólag a számítógéppel támogatott tervezési módszerek válnak be. A Xilinx cég megfizethető áron igen komoly számítógépes fejlesztőeszközöket szállít az automatizált tervezéshez. A Xilinx fejlesztőrendszer programjai PC/AT-ken, AT/386/486-osokon, valamint RISC-processzoros munkaállomásokon futtathatók. A rendszer nyitott felépítésű, több elterjedt és népszerű mérnöki tervező munkahely kimenetét kezeli. Menüvezérelt, gyors és hatékony: a tervezési folyamat lépéseit logikus sorrendben tárja a felhasználó elé. A terv bevihető grafikusan: kapcsolásirajz-szerkesztővel, hagyományosan: TTL vagy hasonló áramköri szimbólumokat használva, valamint logikai egyenletek segítségével. Az automatizált blokkelhelyező és -huzalozó szolgáltatás megkönnyíti az egyszerűbb feladatok gyors megtervezését. A rendszer szimulációs programjával a logikai és a valós idejű, az áramkörön belüli fizikai késleltetéseket is figyelembe vevő szimuláció a tényleges beprogramozás előtt elvégezhető.

Tervezői keretrendszer

A tervezőrendszer programjait és szolgáltatásait összefogó keretrendszer a Xilinx Design Manager (XDM). A főmenü tükrözi a tervezés menetét (Tervbevitel, Fordítás, Elhelyezés és huzalozás, Ellenőrzés). Az első almenüben a számítógépen installált grafikus szerkesztőprogramok nevei láthatók. A tervezők egyidejűleg használhatják a számukra leginkább megfelelő szerkesztőprogramot. A Fordítás almenü a különböző tervbeviteli programok kimeneteit egységes formátumra alakító programokat, a tervezési szabályokat ellenőrző programot, a particionáló programot, valamint további segédeszközöket kínál. A harmadik almenü az automatikus (APR) programot és a manuális szerkesztőprogramot, az Ellenőrzés almenü pedig a szimulációhoz, az in-circuit emulációhoz és a konfiguráláshoz szükséges programokat tartalmazza.

Tervbevitel

Mivel számos kiváló grafikus áramkörtervező CAD rendszer létezik, a Xilinx saját kapcsolásirajz-szerkesztő programot nem készített, csupán az illeszkedést biztosította ezekhez (FutureNet DASH, PCAD, VIEWLogic, OrCAD, CASE és egyebek). A tervbeviteli programokhoz a Xilinx szimbólumkönyvtárában szereplő makrók és TTL áramköri jelölések használhatók a kapcsolási rajz megszerkesztéséhez. A tervezőnek nem kell egy teljesen új, szokatlan jelölésrendszert megtanulnia, kamatoztathatja TTL áramköri ismereteit, jól megszokott kapcsolásirajz-szerkesztő programját. Előnyös ez akkor is, amikor egy régebben tervezett áramkört kell korszerűbben megvalósítani.

A kapcsolási rajzot a fejlesztőrendszer által értelmezhető formátumra kell alakítani. Az egységesített leíró szerkezet neve XNF, Xilinx Netlist Format. Segítségével nemcsak grafikus szimbólumokkal megadott rajz, hanem a szöveges formátumú terv is ábrázolható. Az áramköri terv egyes részletei logikai egyenletek formájában, szöveges leírással is megadhatók. A szöveges állományban vannak a szimbólumon belüli kombinációs vagy logikai hálózat leíró egyenletei, ezeket egy megfelelő program alakítja XNF formátumra. A grafikus és szöveges terv különálló XNF állományai egyetlen, a teljes tervet képező XNF állománnyá egyesíthetők.

Megvalósítás

A tervbevitel után az áramkör blokkokra tördelése (particionálása), a blokkok elhelyezése és behuzalozása, valamint a konfigurációs bitminta előállítása következik. A tördelés automatikus, a további folyamatok az interaktív XACT szerkesztőprogrammal manuálisak vagy az APR (Automatic Place and Route) programmal automatikusak lehetnek. A megvalósítási folyamat nagyágyúja az XMAKE program. Ez előállít egy ún. MAKE fájlt: sorban, hierarchikus felépítésben tartalmazza az elvégzendő tennivalókat, az állományok függőségi viszonyait, majd ennek megfelelően végrehajtja a tervezési folyamat lépéseit.

Az adatbevitel során a kapcsolási rajzon is megadhatók előírások, amelyekkel a tördelő és az APR program működése befolyásolható. Az APR felhasználói beavatkozás nélkül is képes különböző változatokat készíteni, ezekből a legmegfelelőbb kiválasztható. Az eredmény ún. LCA-fájlformátumban keletkezik, és tartalmazza az elhelyezési és huzalozási információkat is.

Az APR tájékoztat tevékenysége eredményéről: időzítési információkról, a huzalozás sorrendjéről, az esetlegesen kihagyott bekötésekről. Az utólagos manuális munkákhoz az XACT szerkesztőprogram való. Ez a program alakítja át az LCA végleges konfigurációs információit egyetlen bináris állománnyá. E fájl tartalma egyenesen a konfiguráló PROM-ba vagy EPROM-ba égethető, vagy a letöltőkábelen keresztül az LCA áramkörbe juttatható.

Ellenőrzés

Ez az utolsó lépés. Legegyszerűbb módja a valós körülmények közti kipróbálás. A számítógép párhuzamos portjára kapcsolt Xilinx letöltőkábellel rendkívül gyorsan átvihető a konfigurációs információ az LCA áramkörbe.

Érdemes az áramkör működését is szimulálni. A szimuláció kétféle lehet: logikai, amely az áramköröknek csak a funkcionális működését ellenőrzi, valamint időhelyes, amely figyelembe veszi a jelterjedési időket is, így kvantitatív képet ad a működésről. A szimulációs program a felhasználó által megadott gerjesztőmintával dolgozik. Az egyes bemenetek statikus szintek, meghatározott időzítésű aszinkron jelek vagy órajelek lehetnek. A szimulátor grafikus formában is kiadja a szimuláció eredményét.

Ha a gondos tervezés és szimuláció ellenére sem megfelelő a működés, a felhasználó kísérletet tehet a hiba kijavítására, majd az áramköri módosítások után átkonfigurálhatja a cellatömböt. Komolyabb zűr esetén az XACTOR In-Circuit Verifier készüléken az LCA belső állapota, tárolóinak tartalma, kivezetései működés közben jeleníthetők meg, megkönnyítve a hiba behatárolását és kijavítását.

Lóth Tamás—Tóth József

A Cédrus Karolina Áruház új nyitvatartási rendje:

Hétfőtől péntekig:
9.00 — 18.00
Szombaton:
9.00 — 13.00

Bp. XI., Karolina út 17.

Unix-fájlok, mint fekete dobozok

Tartalom és jog

Sorozatunk előző részében olyan Unix-parancsokat ismertettünk, amelyek segítségével könyvtár- és fájlkezelő műveletek végezhetők. Ezekkel az utasításokkal többek között fájlokat lehetett másolni, átnevezni, tartalmukat a képernyőre íratni stb. Most olyan parancsokat fogunk bemutatni, amelyekkel a fájlok tartalma vizsgálható. Ismertetünk még néhány segédprogramot is, valamint a fájlokhoz rendelhető elérési jogokat. Szeretnénk felhívni a figyelmet arra, hogy az utasításoknak általában csak a legegyszerűbb formáját tárgyaljuk. A módosító opciók és azok hatásai a kézikönyvekben megtalálhatók.

Eddigi ismereteink szerint a fájlok tartalma vagy a cat, vagy a more utasítással íratható a terminálra. A cat folyamatosan listáz, míg a more egyszerre csak egy képernyőnyi információt jelenít meg. Mind a kettő végigmegy a teljes fájlon. Sokszor előfordul azonban, hogy egy fájlnak csak az első vagy az utolsó néhány sorára vagyunk kíváncsiak. A Unix erre is kínál megoldást. A head és a tail utasítás egy fájl első és utolsó sorait listázza. Opcióként megadható a sorok száma. Ennek hiányában, alapértelmezésként tíz sor jelenik meg a képernyőn. Például a

head fájl_név

az adott fájl első tíz sorát, a

tail – 20 fájl_név

a paraméterként szereplő fájl utolsó húsz sorát listázza.

Műveletek szövegfájlokkal

A Unix olyan szolgáltatásokat is tartalmaz, amelyek segítségével a szöveges fájlokban lévő adatok vizsgálhatók. Ezekkel a programokkal a fájlok tartalma összehasonlítható, sorba rendezhető, a fájlban szövegrészek kereshetők, továbbá megszámlálhatók a fájlban található szavak és sorok.

Két fájl összehasonlítására a diff program alkalmas. Eredményeként azok a sorok kerülnek a képernyőre, amelyek a két fájlban nem egyformák. Használata:

diff fájl_név1 fájl_név2

A fájlok tartalma a sort utasítás segítségével sorba állítható. A legegyszerűbb esetben egy fájl tartalmát a sorokban található első betű alapján rendezi abécé szerint, és megjeleníti a képernyőn. Ezt a listát a már ismert > operátor segítségével fájlba is irányíthatjuk. A sort parancs a rendezendő fájl tartalmát nem változtatja meg! Példák:

sort fájl_név

sort nevek > névsor

Az utasítás pipe-on keresztül szűrőként viselkedik. Ilyenkor egy másik program kimenőadatait rendezi. Például a who paranccsal megtudható, hogy pillanatnyilag kik használják a rendszert. Ha ezt névsorban szeretnénk látni, a következő parancsot kell begépelni:

who | sort

Néha szükség lehet olyan információra is, hogy egy fájl hány sorból, hány szóból vagy hány karakterből áll. A Unixnak erre is van segédprogramja. A wc (word count) utasítással ezek a paraméterek megtudhatók. Ha opció nélkül, csak a fájlnév paraméterrel adjuk, akkor sorok, szavak, karakterek sorrendben mind a három értéket megkapjuk. A – l (line), – w (word), – c (character) módosítók használatával külön-külön is megkaphatjuk a kívánt darabszámot. Az alábbi példákat mutatjuk:

wc fájl_név

wc – l fájl_név

wc – wc fájl_név

Ha a fájl_név paraméter nem szerepel, a bemenőadatokat a standard inputról (stdin), azaz a klaviatúráról várja. Szűrőként is funkcionál. A következő utasítással például könnyen megkaphatjuk, hogy hányan dolgoznak a rendszerrel.

who | wc – l

Fontos szolgáltatása a Unixnak a grep program, amelynek segítségével fájlokban szövegminták kereshetők. A következőképpen:

grep minta fájl_név

A mintát a paraméterként megadott fájlokban keresi. Ha egy fájlban van a mintával megegyező szövegrész, a fájl neve és a mintát tartalmazó sor megjelenik képernyőn. Az utasítás hatása sok kapcsolóval módosítható. Ezek közül most csak kettőt ismertetünk. Ha a programot a következő formában használjuk:

grep – l minta fájl_név

akkor a mintát tartalmazó fájloknak csak a nevét kapjuk meg. Ha arra vagyunk kíváncsiak, hogy mely sorok tartalmazzák a keresett részt, a – n opciót kell jelölnünk. Például a

grep – n Gabor /etc/*

utasítás a /etc könyvtárban minden fájlt megvizsgál. Ha olyat talál, amelyben a Gabor név szerepel, kiírja a fájl nevét, a nevet tartalmazó sor sorszámát és magát a sort. Az utasítás kapcsán ügyeljünk arra, hogy a több szóból álló mintát idézőjelek közé kell írni.

Oszlopos tudnivalók

Az eddigi utasítások a szövegfájlok soraival foglalkoztak. Most egy olyan eszközt mutatunk be, amellyel a fájlok soraiból „kiollózhatjuk” azokat az oszlopokat, amelyekre kíváncsiak vagyunk. Oszlop alatt valamilyen adott jellel elválasztott mezőket vagy a sorban található karaktereket értjük. Az utasítás neve cut, és az oszlopfajtáknak megfelelően kétféle formában használható:

cut – f1,5 – d: /etc/passwd

cut – c1– 12,45– /etc/group

Az első példa az /etc/passwd fájl :-tal elválasztott mezői közül az elsőt és az ötödiket listázza ki a képernyőn. A mezők sorszámát a – f kapcsoló után kell megadni, míg a – d után a mezőket elválasztó (szeparátor) jel adható meg. A második példa karakterekre vonatkozik, amit a – c kapcsoló jelez. Az /etc/group fájl sorainak az első 12 és a 45-ik pozíciótól kezdődő karaktereit fogja listázni. Az utasításnak szűrő hatása is van: például a következő utasítás a rendszert használóknak csak a nevét fogja kiírni.

who | cut – c1– 8

Fájlvédelem

Az előző részben az ls – l parancs ismertetésénél már szó volt az ún. fájlhozzáférési jogokról. Vizsgáljuk meg ezt a témát kicsit részletesebben. A Unixban a fájlok és könyvtárak elérése engedélyekhez köthető. Mivel a Unix többfelhasználós operációs rendszer, szükség van a az ilyen jellegű védelemre. Az elérési jogokat a fájl tulajdonosa adhatja meg. Ezeket az ls – l parancs végrehajtása után megjelenő lista első tíz karaktere jelzi. Az első karakter a fájl típusát adja meg, a maradék pedig hármas csoportokban a hozzáférési jogokat.

A fájl típusánál számunkra jelenleg csak a közönséges fájlok és a könyvtárak érdekesek. A közönséges fájloknak (mivel ezekből van a legtöbb) nincs külön betűjele. Ezeknél egy - karakter (mínuszjel) áll a fájltípus helyén. A könyvtárakat egy d betű (directory) azonosítja.

A háromkarakteres hozzáférési kódok, amelyek megadják, hogy egy fájl olvasható, írható vagy végrehajtható, sorrendben a következő felhasználókra vonatkoznak:

— a fájl tulajdonosára,
— a tulajdonos csoportjára,
— minden más felhasználóra.

A hozzáférési kódban sorrendben az rwx betűk szerepelhetnek. Ha mind a három látható, a fájl olvasható, írható és végrehajtható is. Ha valamelyik akció tiltva van, akkor a megfelelő betű helyén egy „-" jel szerepel. Könyvtáraknál az olvasás a fájlok listázását, az írás a fájlok létrehozását, illetőleg törlését jelenti. Egy könyvtárhoz az 'x' végrehajtási engedély nélkül nem lehet hozzáférni, hiába van például az olvasása engedélyezve. Ilyenkor a cd utasítással sem lehet odajutni. Az elmondottak szemléltetésére nézzünk most két példát.

– rwx – – x – – –

Ez a kód egy közönséges fájlt jelöl, amelyet a tulajdonosa olvashat, írhat és végrehajthat. A tulajdonos csoportjának csak végrehajtási engedélye van, míg a többi felhasználó semmilyen formában nem férhet hozzá.

drwxr – x– – x

Itt egy könyvtárról van szó, amelyhez a tulajdonosa korlátozás nélkül hozzáférhet. A tulajdonos csoportja elérheti a könyvtár fájljait, listázhatja is azokat, de nem módosíthatja a katalógus tartalmát. A többi felhasználó a cd utasítással eljuthat a könyvtárba, de az olvasási engedély hiányában nem tudhatja meg (nem listázhatja) a tartalmát. Használhatja viszont az ott található fájlokat, ha ismeri a nevüket.

A kirekesztés módozatai

A fájlhozzáférési kódokat a chmod utasítással lehet megváltoztatni:

chmod kód fájl_név

A kódot számokkal vagy betűkkel lehet leírni. Számok esetén a három csoportnak megfelelően három, 0 és 7 közé eső számot kell megadni. A nulla a három elérési mód tiltását, a hét mind a háromnak az engedélyezését jelenti. Betűkkel érthetőbben írható le a kód. Ilyenkor meg kell adni, hogy melyik felhasználócsoport hozzáférési jogait hogyan akarjuk módosítani. A fájl tulajdonosát az 'u' (user), a tulajdonos csoportját a 'g' (group), míg az összes többi felhasználót az 'o' (other) betű azonosítja. Ezek után lássunk néhány konkrét példát.

chmod u+x testfile

Ez az utasítás a testfile nevű fájl végrehajtását engedélyezi a fájl tulajdonosának. A következő parancs a tulajdonos csoportjának és a többi felhasználónak a testfile írását és végrehajtását engedélyezi.

chmod go+wx testfile

Végül nézzünk meg egy tiltást, ahol az ún. összes többi felhasználó valamennyi hozzáférési jogát megszüntetjük.

chmod o – rwx testfile

Még néhány szolgáltatás

Az eddigieken kívül szeretnénk még néhány hasznos segédprogramot ismertetni. Az első ezek közül a find parancs, amely fájlok keresésére használható. Leggyakoribb formája:

find path_név – name fájl_név – print

A path_név annak a könyvtárnak a neve, ahol a keresést kezdjük. A program innen indulva az összes alkönyvtárat végigjárja. A keresett fájl nevét a – name opció után kell megadni. A – print kapcsoló hatására a megtalált fájl neve a képernyőre íródik.

A következő két szolgáltatás segítségével a dátumot és az időt, valamint egy tetszőleges év tetszőleges hónapjának naptárját lehet megkapni. A dátum és az idő a date utasítás hatására íródik a képernyőre. A

cal feb 1952

paranccsal pedig megtudhatjuk, hogy milyen napra esett például 1952. február 13.

Végül megemlítjük a bc interaktív kalkulátor segédprogramot, amellyel a szokásos műveletek gyorsan és pontosan elvégezhetők.

Déri Gábor

E számunk hirdetői

	Info#	Oldal
Barex	23	55.
CeBIT	16	27.
Computerland	03	B4.
Controll	05	60.
Cédrus Kiadó	30	20.
Cédrus Rt. (Polaroid)	28	21.
Data Doctor	22	56.
Datentechnik	19	24.
FAN Computer	09	56.
Floppyland	24	K4.
Galax	20	K4.
Interag (Mitac)	01	B2.
IR Szerviz	06	59.
Keszo	25	K4.
Macroda	08	56.
Made-Info	26	30.
Magics	07	58.
NTT 2000	04	60.
Presentex (CeBIT)	29	56.
Qwerty	13	55.
Szoftver ABC	12	31.
Szolinfo	27	31.
Toner	14	31.
Trendex	11	36.
Unitrade	15	55.
Userland (Remind)	02	B3.
Vénusz	17	55.
Xenon	10	36.
Xfer	21	23.

A Unix shell programozása II.

Adalékok a „recepthez”

A sorozatot azoknak az eszközöknek a bemutatásával folytatjuk, amelyek a programozásban kicsit is jártas olvasóinknak már ismerősek lesznek — ezekre így-úgy, de mindenképpen szükség van a munkához. Nem jutunk el most az összes ilyesmi felvonultatásáig, viszont ami késik, nem múlik...

Bizonyos előre definiált shellváltozókat a shell használ, ezek közül néhányat a rendszer bejelentkezéskor iniciál. Ezek értékét is tudja a felhasználó változtatni. Közülük a két legfontosabb a HOME változó — amely a felhasználó alapmunkakatalógusát (home directory) tartalmazza — és a PATH változó, amely kettőspontokkal elválasztva azokat a katalógusokat sorolja fel, ahol a rendszer a kiadott parancsokat keresi. (Csak akkor, ha nem teljes nevet adunk meg; ha az a parancs, hogy /bin/date, akkor nincs mit keresni.) Ponttal vagy üres stringgel az aktuális munkakatalógust lehet jelölni. Ha például a PATH változó tartalma:

```
:/bin:/usr/bin:/etc
```

és kiadjuk a comfile parancsot, akkor a shell először az aktuális munkakatalógusban keres egy comfile nevű végrehajtható fájlt, ha nem talál, akkor a /bin alatt folytatja a keresést, ha itt sincs, akkor a /usr/bin-nel, végül a /etc-vel próbálkozik. Más előre definiált változónevek értékét is a shell tartja karban. Ilyen a már említett $# változó (a pozicionális paraméterek számát tartalmazza), a $$ változó, amely az éppen futó process azonosítóját tartalmazza (ez gyakori egyedi fájlnevek előállításánál), a $! változó, amely a háttérben utoljára elindított folyamat azonosítóját adja meg; és még két változó, amelyekről csak a későbbiekben lesz szó.

Programok visszatérő értéke

Normálisan a programok 0 visszatérő értéket szolgáltatnak. Az exit n utasítással lehet a program futását úgy befejezni, hogy a visszatérő érték n legyen. Az utoljára végrehajtott parancs visszatérő értékét a $? shellváltozó tartalmazza:

```
comfile tartalma:
comm2
echo $?
comm2 tartalma:
exit 23
Kimenet:
23
```

Ha paraméter nélkül adjuk az exit utasítást, az utoljára végrehajtott parancs visszatérő értékével jön vissza a program.

Az if utasítás (1)

A feltételes utasítás legegyszerűbb alakja a következő:

```
if utasítás1
then utasítások
fi
```

Az if utasítások egymásba ágyazhatók — a lezáró fi-vel ellátva. Ha utasítás1 visszatérő értéke 0, akkor végrehajtódnak a thent követő utasítások, egyébként nem. (Ez az egyik olyan eltérés a shell és a C nyelv között, ami zavaró lehet: a shell számára az „igaz” értéket 0 képviseli, a „hamis”-at pedig minden más; a C-ben ez éppen fordítva van.) Például:

```
comfile tartalma:
if comm2
then
echo If!!!
fi
comm2 tartalma:
exit 0
Kimenet:
If!!!
```

Az if utasítás else-ággal is kiegészíthető:

```
if utasítás1
then if-utasítások
else else-utasítások
fi
```

Például:

```
comfile tartalma:
if comm2
then
  echo If!!!
else
  echo Else!!!
fi
comm2 tartalma:
exit 39
Kimenet:
Else!!!
```

Az if, fi, else kulcsszavaknak a sorban az első helyen kell állniuk. Szintaktikailag hibás például:

```
if comm2 then echo If!!!
```

(Tudniillik lehet a comm2 parancsnak egy olyan paramétere, hogy 'then'!)

A test utasítás

Az if utasítás leggyakrabban a test utasítással együtt szerepel. A test-tel stringeket, numerikus értékeket és fájltípusokat lehet megvizsgálni; most csak a „numerikus” példát mutatjuk be.

test op1 operátor op2

Ha a vizsgálat eredménye igaz, a test utasítás visszatérő értéke 0, különben más. Az operátorok lehetnek:

- –eq egyenlő (equal)
- –ne nem egyenlő (not equal)
- –gt nagyobb (greater than)
- –lt kisebb (less than)
- –ge nagyobb vagy egyenlő (greater than or equal to)
- –le kisebb vagy egyenlő (less than or equal to)

E programrészlet a paraméterek számát vizsgálja:

```
if test $# –ne 2
then
  echo Hiba!!!!!! >&2
  exit 1
fi
```

A shell a nem numerikus változók vagy paraméterek értékét 0-nak tekinti. A test utasításnak van egy tömörebb (és nehezebben olvasható, de elterjedtebb) alakja is: a megvizsgálandó kifejezést szögletes zárójelek közé kell tenni úgy, hogy a kezdő zárójel után és a bezáró zárójel előtt kötelező legalább egy szóközt tenni:

```
if [ $# –ne 2 ]
then
  echo Hiba!!!!!! >&2
```

```
  exit 1
fi
```

A szóközökre azért van szükség, mert egy string tartalmazhat „[" vagy „]" karaktereket is:

```
X=[abcd]efgh
```

Szóköz nélkül a shell nem ismeri fel, hogy itt a [jelnek speciális jelentése van. Egy kicsit összetettebb alkalmazási példaként nézzük, amikor a bejelentkezett felhasználók számától függ, hogy hajlandók vagyunk-e elindítani egy programot:

```
usernum='who | wc – l'
if [ usernum – gt 6 ]
then
  echo Tilos!
  exit 1
fi
```

Fájlnévleíró metakarakterek

A shell ismer néhány metakaraktert, amelyek fájlnévcsoportok kijelölésére alkalmasak:

- * Tetszőleges karakterek tetszőleges hosszúságú sorozata (0 hosszt is beleértve).
- ? Egyetlen tetszőleges karakter.
- [...] A zárójelen belül megadott karakterek bármelyikének egyetlen előfordulása.
- – A kötőjellel elválasztott karakterpár egy intervallumot jelöl.
- ! A szögletes zárójelen belüli első pozícióban negáló funkciója van: ekkor a fel nem sorolt karakterek fognak megfelelni a mintának.

Például:

A*B	Minden A-val kezdődő és B-re végződő string. (AB, ACCCB, ...)
dump	Minden olyan string, amelyben dump bárhol előfordul. (dump, aaadump, aaadumpbbb,...)
A?	Az összes A-val kezdődő 2 karakteres string.
/usr/bin/???	Az összes 3 karakteres név a /usr/bin katalógusban.
[a-zA-Z]	Az összes kis- és nagybetűkből álló 1 karakteres string.
[b-emno]	A b, c, d, e, m, n, o betűkből álló 1 karakteres stringek.
[!0-9]	A nem számjegyből álló 1 karakteres stringek.
[a-z]*[0-9]	Kisbetűvel kezdődő, számjegyre végződő stringek.

A szövegfeldolgozó programok meta-karakterkészlete ettől eltér! Pontosabban fogalmazva: egyes karaktereket más jelentéssel használnak a reguláris kifejezéseket feldolgozó programok (grep, ed, sed, awk).

A case utasítás

A case utasítás legegyszerűbb formája a következő:

```
case string in
        minta)  utasítások ;;
        minta)  utasítások ;;
        ...
        minta)  utasítások ;.
esac
```

A minták ugyanazokkal a metakarakterekkel adhatók meg, mint a fájlnévleírás. A shell sorban összehasonlítja a stringet az adott mintákkal; amikor először egyezést talál, végrehajtja a megfelelő utasításokat. Hangsúlyozzuk, hogy az összehasonlítás sorban halad, tehát ha valamelyik minta helyére *-ot írunk, akkor a következő minták soha nem lesznek kiértékelve. (A * felel meg a C nyelvben használatos switch utasítás default ágának, csak éppen ott a sorrend közömbös! Vegyük észre viszont, hogy itt stringeket, sőt metakaraktereket is meg tudunk adni, míg a C-ben csak numerikus konstansokat!) Nézzünk egy programrészletet, amelyben beolvasunk egy stringet és attól függően csinálunk mást-mást, hogy igen, nem, vagy egyéb választ kaptunk:

```
read answer
case $answer in
  [Ii]gen)  echo Hát igen?
  ;;
  [Nn]em )  echo Hát nem?
  ;;
  *         ) echo Mi?????
  ;;
esac
```

A case utasítás egyes ágaiban több mintát is megadhatunk | jellel elválasztva, ekkor bármelyik mintával egyezéskor az ide tartozó utasításokat hajtja a shell végre. Például:

```
read answer
case $answer in
  [Ii]gen | [Yy]es)  echo Hát igen?
  ;;
  [Nn]em  | [Nn]o )  echo Hát nem?
  ;;
  *                ) echo Mi?????
  ;;
esac
```

A while ciklus

A while utasítás szintaxisa:

```
while parancslista
do
  utasítások
done
```

A parancslista végrehajtódik; ha az utolsó parancs visszatérő értéke 0, végrehajtódnak a ciklustörzs utasításai, egyébként a következő programsorra adódik a vezérlés. A while, do, done kulcsszavaknak a sor első szavának kell lenniük, csakúgy, mint az if, then, else, fi kulcsszavaknak, valamint a többi kulcsszónak. (A szabály pontosabb megfogalmazását lásd később.) A ciklusból kiugrás eszköze a break utasítás. Az alábbi programrészlet igenlő vagy tagadó választ kér a felhasználótól:

```
echo Kéri a dátumot \(i/n\)?
while read answer
do
 case $answer in
   [Ii]*) echo Akkor hát küldöm.
      date
      break
      ;;
   [Nn]*) echo Ha nem, hát nem.
      break
      ;;
   *      ) echo Hibás válasz!
      ;;
 esac
done
```

Figyeljük meg, hogy a zárójelet escape karakterrel kellett megvédeni, mivel ez is speciális jelentéssel bír a shell számára (lásd később). A read utasítás mindaddig 0 visszatérő értékű, míg fájlvégjelet nem kap (a felhasználó control–d karaktert nem küldött). A true vagy a false kulcsszavakkal végtelen ciklusok alakíthatók (while true).

Nemes Mihály

Prix Ars Electronica 92

A világ legismertebb számítógépes művészeti pályázata és kiállítása.
Bárki pályázhat számítógépes grafikával, animációval, zenével és interaktív művészeti alkotásokkal.

Összdíjazás: 1 250 000 schilling.

Beküldési határidő: 1992. február 29.
Cím: ORF – Prix Ars Electronica Franckstrasse 2a, A-4010 Linz, Austria

Modula-2

Egyebek...

Most a Modula-2 állandóival, adattípusaival, változóival és műveleteivel kapcsolatos fogalmakat mutatjuk be. A prezentálás meglehetősen formális, de a szintaktikailag helyes programok írásához és a nyelvi szolgáltatások maximális kihasználásához kellenek az alapos információk. A mágneslemez mellékleten az LST fájl a száraz definíciókat példákkal illusztrálja.

Programjainkban gyakori követelmény, hogy az adott típusok egymással kompatibilisek legyenek. A kompatibilitási osztályok (csökkenő erősségben):

Típusazonosság — A legszigorúbb követelmény. Két típus azonos, ha azonosítójuk ugyanaz, vagy ha az egyik típust a másik típus azonosítójával deklaráltuk. Eljáráshíváskor változóparaméter esetén az aktuális és a formális paraméter típusának azonosnak kell lennie. Ezt magyarázza például azt a tény, hogy az alaptípusokra vonatkozó beviteli eljárások (ReadCard, ReadInt, Read stb.) nem használhatók az alaptípusokból származtatott, intervallum típusú változók beolvasására.

Típuskompatibilitás — Két típus kompatibilis, ha egyik a másik intervallumtípusa, ha mindkét típus ugyanazon típus intervallumtípusa, vagy ha a két típus azonos. Az ADDRESS típus minden mutatótípussal kompatibilis, a SHORTADDR minden bázisos mutatóval. A típuskompatibilitás kifejezések és relációk kiértékelésénél szükséges.

Értékadási kompatibilitás — A fentieken kívül az alábbi típuspárok értékadás-kompatibilisek egymással: CARDINAL / INTEGER, SHORTCARD / SHORTINT, LONGCARD / LONGINT.

A Modula-2 előre definiált WORD típusa egy gépi szónyi memóriaterületnek felel meg. A standard Modula értelmezése szerint a WORD mérete 8 bites processzor esetén 1 bájt, 16 bitesnél 2 bájt, 32 bitesnél 4 bájt, illetve nem bájtorientált proceszoroknál például akár 4, 9, 21 vagy 24 bites gépi szó. A TopSpeed Modula-2 két másik hasonló célú előre definiált típussal, a BYTE és a LONGWORD típussal rendelkezik. A három által reprezentált memóriaterület az IBM PC-n 1 bájt (BYTE), 2 bájt (WORD) és 4 bájt (LONGWORD). Ezek minden azonos méretű típussal értékadás-kompatibilisek. Az ISO-szabvány LOC típusa az adott processzor által megcímezhető legkisebb memóriarekeszt jelenti, így a LOC a PC-n 1 bájtnak felel meg. (A BYTE standard típusra tett javaslatot — a nem bájtorientált implementációkra tekintettel — elvetették, helyette született meg a LOC.)

Objektumok és értékek

Az objektumok kizárólag a típusuknak megfelelő értékeket vehetnek fel. (Ebben az összefüggésben nem az OPL-ekkel kapcsolatos általánosabb objektumról van szó.) A típus határozza meg az objektum tulajdonságait, azaz az objektum által elfoglalt memóriaterületen található bitminta értelmezését. Az objektumoknak két osztálya van: a változók és a formális paraméterek. A változókat, mielőtt hivatkoznánk rájuk, deklarálni kell. A változókhoz a fordító egy tárterületet rendel, melynek méretét a változó típusa szabja meg, és mindig a változó aktuális értékét tartalmazza. A fordítás során a fordító a változó azonosítójára való hivatkozást a forráskódban, a memória megfelelő területére való hivatkozással (címmel) helyettesíti a tárgykódban. A változók a program adatainak tárolására szolgálnak. A rekord és a tömb típusú objektumoknak több komponens objektumuk lehet.

```
$ Deklaráció = VAR {
VáltozóAzonosító { ","
VáltozóAzonosító } ":" TípusDefiníció }.
$ VáltozóAzonosító = Azonosító    [
"[" Kifejezés ":" Kifejezés "]" ].
```

A deklarációs listában szereplő változók adott típusúak lesznek. A változók kezdőértékei meghatározatlanok. Mint már korábban láttuk, a változók deklarálása (pontosabban az azonosítók deklarálása) az adott blokkra, illetve az ezen belül elhelyezkedő blokkokra érvényes. Ha egy belső blokkban egy azonosítót újra deklarálunk, akkor a belső blokkban deklarált változóazonosítóval erre az ún. lokális változóra hivatkozunk. A változók a deklarációjukat tartalmazó blokkokkal együtt szűnnek meg. Változókat tetszőlegesen összetett típusúaknak is deklarálhatunk, külön típusdefinícó nélkül, ezek az ún. anonim típusok. Az anonim típusok használata szintaktikailag ugyan helyes, programozástechnikai szempontból azonban, mégsem ajánlható.

Változókat fix fizikai címre is helyezhetünk. Ehhez IBM PC esetén két konstans CARDINAL kifejezéssel meg kell adnunk a szegmens és offszet értékeket. Lineáris cím esetén természetesen egyetlen kifejezés határozza meg a címet.

Állandók

Az állandók az alapvető konstansértékeket jelölik a forráskódban. A számkonstansok decimális egész, hexadecimális egész, oktális egész, illetve valós számok lehetnek. A szövegkonstansok (karakterláncok) aposztrófok vagy idézőjelek közé írt ASCII karakterek vagy oktális karakterkonstansok lehetnek.

```
$ Érték = EgészSzám | ValósSzám |
Karakterlánc.
```

Az egész számok az egész- és pozitívegész-típusok, a valós számok a valósszámtípusok, míg a karakterláncok a karaktertípusok (ha a karakterlánc hossza 1), illetve a karaktertömb-típusok számára definiálnak lehetséges értékeket. Az állandó rekord- és tömbértékeket csoportosítva adjuk meg.

```
$ Érték = Név "(" Kifejezés ","
Kifejezés { "," Kifejezés } ")".
```

A kifejezések, amelyekből legalább kettőnek kell lenni, vagy állandóknak, vagy eljárásazonosítóknak és a névvel megnevezett típusnak a komponensértékeit szolgáltatják. Tömbtípus esetén minden indexre értéknek, rekordtípus esetén az összes mezőnek értéket kell adnunk. Variáns rekord esetén még a hiányzó szelektormezőnek is értéket kell adnunk, hogy meghatározható legyen, melyik variáns érvényes.

Nevesített állandókat is létrehozhatunk, így olyan azonosítókat teremtünk, amelyek állandó értéket képviselnek.
$ Deklaráció = CONST { Azonosító "=" Kifejezés ";" }.

Az előre definiált NIL NearNIL és FarNIL állandók minden (azonos szerkezetű) mutatótípussal kompatibilisek. A NIL értéke a memóriamodelltől függően NearNIL vagy FarNIL. (A NearNIL és FarNIL standard konstansok a TopSpeed Modula bővítései, míg a memóriamodell a IBM PC sajátossága, ezek tehát implementációspecifikusak.)

Sem az állandók, sem a nevesített állandók nem objektumok.

Halmazértékek

Halmazértékeket halmazkonstruktor segítségével hozhatunk létre.
$ Érték = [Név] "{" [VálasztásiLista] "}".

A választások kifejezéseinek típusa megegyezik a halmaz elemeinek típusával, és nem szükséges, hogy állandó kifejezések legyenek. A "Név" a halmaz típusát jelöli, elhagyása a BITSET-et.

Megnevezések

A megnevezéseket objektumok elérésére használjuk. Egy összetett típus komponenseire megnevezésének kiegészítésével hivatkozhatunk. A kiegészítés nevesített, csoportosított állandók komponenseinek azonosítására is használatos. A megnevezés érték is lehet:
$ Érték = Megnevezés.

A megnevezés legegyszerűbb formája az egység neve.
$ Megnevezés = Név.

Ugyanígy használjuk a felsorolt típus értékeinek azonosítóit és a nevesített állandókat.

Az indexelés a tömb típusú objektumok komponenseinek megnevezésére szolgál.
$ Megnevezés = Megnevezés "[" Kifejezés { "," Kifejezés } "]".

Az indexkifejezések felsorolása ugyanannyit tesz, mintha az indexek külön-külön vannak felsorolva: x[i,j,k] megegyezik x[i][j][k]-val. Az indexkifejezésnek értékadás-kompatibilisnek kell lennie az indextípussal, és az indexnek megfelelő komponens objektumot jelöli ki.

A mezőkiválasztás a rekord típusú objektumok komponenseinek megnevezésére szolgál.
$ Megnevezés = Megnevezés "." Azonosító.

Az azonosító a rekordtípus egyik mezőjének azonosítója. A kapott megnevezés a rekord objektumnak erre a mezőjére vonatkozik. A mutató típusú objektumok által mutatott objektumokra a megnevezés által hivatkozhatunk.
$ Megnevezés = Megnevezés "^".

Ha a mutatótípus bázisos, a megnevezés egyben a báziskifejezés kiértékelését is jelenti. Az indexelés, a mezőkijelölés és a hivatkozás tetszőlegesen kombinálható, és függvényhívásokat is tartalmazhat. A TopSpeed Modula-2 mutatókonstruktora lehetővé teszi CARDINAL típusú szegmens- és offszetkifejezések kombinálását fizikai címmé.
$ Megnevezés = "[" Kifejezés ":" Kifejezés [Név] "]".

Név az eredményül kapott abszolút (szegmens és offszet) mutató típusát adja. Alapértelmezés: FarADRESS típus.

Kifejezések

A kifejezés operandusoknak operátorokkal összekapcsolt sorozata értékek számítását írja le. A kifejezésekben az operátorok által operandusokat kombinálunk, amelyek maguk is lehetnek kifejezések. Az operátorok lehetnek műveleti jelek, relációk és zárójelek. A kifejezéseknek — akárcsak az értékeknek — típusuk van, kivéve ha minden operandus szám. Az ilyen kifejezések kiértékelésekor a típus csak akkor lesz meghatározva, ha a fordításkor a programnak egy olyan részében vannak, ahol erre szükség van.

Amikor egy operátor operandusai állandók, a kifejezés kiértékelése fordítási időben zajlik, és az eredmény is állandó.

Az operátorok prioritása szabja meg egy kifejezésen belül a kiértékelés sorrendjét: a nagyobb prioritású operátorok végrehajtása után következnek a kisebb prioritásúak. Ha az operátorok prioritása megegyezik, a kiértékelés balról jobbra halad. Az operátorok négy prioritási osztályba tartoznak, ezek (csökkenő prioritási sorrendben): unáris, multiplikatív, additív és relációs operátorok. A prioritás és a balról jobbra irányuló kiértékelés módosítható zárójelek segítségével. Ilyenkor a kifejezés kiértékelése a legbelső zárójelpáron belül kezdődik és kifelé halad. A zárójelpárok által a műveletek tetszőleges végrehajtási sorrendjét írhatjuk elő. A zárójelpárokat csoportosításra is használhatjuk, ekkor az elsődleges cél nem a kiértékelési sorrend megváltoztatása, hanem a kifejezés egyes részeinek logikai vagy más szempont szerinti összetartozásának hangsúlyozása.

$ Kifejezés = EgyszerűKifejezés [RelációsOperátor EgyszerűKifejezés].
$ EgyszerűKifejezés = [ElőjelOperátor] Term {AdditívOperátor Term}.
$ Term = Faktor {MultiplikatívOperátor Faktor}.
$ Faktor = "(" Kifejezés ")" | NOT Faktor | Érték.
$ ElőjelOperátor = "+" | "–".
$ MultiplikatívOperátor = "*" | "/" | DIV | MOD | AND | "<" | ">".
$ AdditívOperátor = "+" | "–" | OR.
$ RelációsOperátor = "=" | "#" | "<>" | "<" | "<=" | ">" | ">=" | IN.

Operátorok

Hogy egy operátor milyen műveletet jelöl ki, az az operátortól és az aktuális operandusoktól függ. Tehát az ismert műveleti jelek különböző operációkban más-mást jelölhetnek.

Az IN kivételével minden operátortípus kompatibilis operandusokat követel meg. A relációs operátorok BOOLEAN típusú eredményt szolgáltatnak, míg a többi operátor az operandusaik típusával megegyező típusút. A „+" előjel a numerikus típusokra van definiálva, de nincs hatása. A „–" operátor egész és valós típusokra van értelmezve, és az operandust negálja. A NOT operátor egy BOOLEAN operandus logikai komplemensét képzi.

Az „=" és „#" ("<>") operátorok minden típusra, tehát a programozó által definiált típusokra is vonatkoznak, és ezek egyenlőségét, illetve egyenlőtlenségét vizsgálják. A „<", „<=", „>" és „>=" operátorok a sorszámozott, valós és mutatótípusokra érvényesek, és a relatív sorrendet fejezik ki. A „<=" és „>=" a halmaztípusokra is definiálva vannak, és ilyenkor részhalmaz-relációt jelölnek. Az IN operátor tartalmazásvizsgálatot végez. Jobb oldali operandusa halmaztípusú, bal oldali operandusa pedig a jobb oldali operandus alaptípusának egy lehetséges értéke. Az eredmény TRUE, ha az első operátor a másodiknak az eleme.

Mivel a halmaztípusokat a Modula-2 bittérképként tárolja, ahol az egyes bitek a halmaz lehetséges elemértékeit — azaz egy elem meglétét a halmazban — jelölik, így a „+", „*", „/" és „–" halmazoperátorok a bitenkénti OR, AND, XOR és AND NOT logikai műveleteknek felelnek meg.

Az OR helyett használható a „|", az AND helyett a „&", a NOT helyett a „~" műveleti jel. A TopSpeed Modula-2-ben lehetőség van a logikai AND, OR és NOT operátorokat bitenkénti műveletekként is alkalmazni az egész típusú kifejezésekben.

Villányi László

Klappolhat a Clipper!

A Clipperrel foglalkozó sorozatunk első négy részében bemutattuk a nyelv előre definiált osztályait és ezek használatának lehetőségeit. Sajnos nincs közvetlen lehetőség újabb osztályok definiálására, de az extend rendszer segítségével — amelyről később lesz szó — azért ez is megoldható. Most a változókezeléssel kapcsolatos ismereteket tekintjük át, különös tekintettel az egyes változók érvényességi körére, valamint a velük végezhető műveletekre.

Az új verzió a változók négyféle hatókörű típusát különbözteti meg.

Az érvényesség változatai

A négy típus fontosabb jellemzői a következők:

A LOCAL változó az őt tartalmazó eljárás meghívásakor keletkezik, és csak ebben az eljárásban érvényes; az eljárásból való visszalépés alkalmával megsemmisül. Ha az eljárás önmagát hívja (vagyis rekurzív), akkor minden egyes alkalommal új változó jön létre.

LOCAL kulcsszó nélkül is LOCAL-ként deklarálódnak a FUNCTION vagy PROCEDURE parancs után zárójelben felsorolt paraméterek.

A STATIC változó ugyancsak az őt tartalmazó eljárás meghívásakor jön létre, de nemcsak ebben, hanem az összes, ebből (közvetlenül vagy közvetve) hívott eljárásban is érvényes. A deklarációt tartalmazó eljárásból visszatérésekor nem semmisül meg, csak láthatatlanná válik. Ha az eljárást újra hívjuk, a változó újra látható lesz, és az utoljára beleírt értéket tartalmazza. Ha a STATIC változót az első eljárás előtt deklaráljuk, a forrásfájl minden eljárásában érvényes lesz. Ez utóbbi esetben fordításkor a /N opciót meg kell adni.

A LOCAL és a STATIC deklarációnak minden egyes végrehajtható utasítást meg kell előznie az eljáráson belül. Makrókifejezésben nem deklarálhatók. Típusukat csak a VALTYPE( ) függvénnyel határozhatjuk meg. Az azonos nevű PRIVATE és PUBLIC változókat elrejtik, azonos nevű FIELD, MEMVAR — továbbá a STATIC mellett a LOCAL, míg a LOCAL-nál a STATIC — változók esetén fatális hiba keletkezik. LOCAL és STATIC változókat nem lehet .MEM kiterjesztésű fájlba menteni vagy onnan előhívni. Ha kezdeti értékadást nem alkalmazunk, a változók kezdőértéke NIL, s ha tömbváltozók, minden elemük NIL lesz.

A PRIVATE változó a deklarálásakor jön létre; a deklarációt tartalmazó minden ebből (közvetlenül vagy közvetve) hívott eljárásban érvényes. A deklarációt tartalmazó eljárásból visszatéréskor semmisül meg. Érvényességi ideje alatt megsemmisíthető a CLEAR ALL, CLEAR MEMORY, RELEASE utasítások akármelyikével.

PRIVATE kulcsszó nélkül is PRIVATE-ként deklarálódnak a PARAMETERS sorban megadott változók, és minden más, nem deklarált változó. A PRIVATE változó elfedi az azonos nevű PUBLIC változót. Ha a PRIVATE változó megsemmisül, akkor a PUBLIC újból láthatóvá válik.

Ha mást meg nem adunk, a PRIVATE változó kezdőértéke is NIL, ha tömbváltozó, minden eleme NIL lesz.

A PRIVATE és PUBLIC változók együttes száma nem haladhatja meg a 2048-at.

A PUBLIC változó szintén a deklarálásakor jön létre, de érvényben marad mindaddig, amíg explicit módon meg nem szüntetjük. Ezt a CLEAR ALL, CLEAR MEMORY, RELEASE utasítások bármelyikével megtehetjük.

A PUBLIC és a PRIVATE is végrehajtható utasítások, így bárhol elhelyezkedhetnek az eljáráson belül. Ha akár a PUBLIC, akár a PRIVATE változó neve megegyezik FIELD, STATIC vagy LOCAL típusú változóéval, fatális hiba keletkezik.

Clipper 5-operátorok

*	Num.	Szorzás
/	Num.	Osztás
+	Dátum	Napok hozzáadása a dátumhoz
	String	Stringek összekapcsolása
	Num.	Összeadás
–	Dátum	Napok kivonása a dátumból
	String	Stringek összekapcsolása szóközök kivágásával
	Num.	Kivonás vagy negatív előjel
++	Dátum	Dátum növelése egy nappal
	Num.	Inkrementálás (+1)
—	Dátum	Dátum csökkentése egy nappal
	Num.	Dekrementálás (–1)
=	Dátum	Igaz, ha a két dátum azonos
	String	Igaz, ha a két string egyenlő
	Num.	Igaz, ha a két szám egyenlő
	Nil	Igaz, ha mindkét érték Nil
==	Dátum	Igaz, ha a két dátum azonos
	String	Igaz, ha a két string hosszra is azonos
	Num.	Igaz, ha a két szám egyenlő
	Nil	Igaz, ha mindkét érték Nil
!=,<>,#	Dátum	Igaz, ha a két dátum nem azonos
	String	Igaz, ha a két string nem azonos
	Num.	Igaz, ha a két szám egyenlő
	Nil	Igaz, ha csak az egyik érték Nil
<	Dátum	Igaz, ha a bal oldali dátum korábbi
	String	Igaz, ha a bal oldali string kisebb
	Num.	Igaz, ha a bal oldali szám kisebb
	Nil	Igaz, ha csak a bal oldali érték Nil
<=	Dátum	Igaz, ha a bal oldali dátum korábbi vagy egyenlő
	String	Igaz, ha a bal oldali string kisebb vagy egyenlő
	Num.	Igaz, ha a bal oldali szám kisebb vagy egyenlő
	Nil	Igaz, ha a bal oldali érték Nil
>	Dátum	Igaz, ha a bal oldali dátum későbbi
	String	Igaz, ha a bal oldali string nagyobb
	Num.	Igaz, ha a bal oldali szám nagyobb
	Nil	Igaz, ha csak a jobb oldali érték Nil
>=	Dátum	Igaz, ha a bal oldali dátum későbbi vagy egyenlő
	String	Igaz, ha a bal oldali string nagyobb vagy egyenlő
	Num.	Igaz, ha a bal oldali szám nagyobb vagy egyenlő
	Nil	Igaz, ha a jobb oldali érték Nil
=,STORE	Dátum	Dátumértékadás
	String	Stringértékadás
	Num.	Numerikus értékadás
	Nil	Nil érték írása változóba
:=	Dátum	Dátum kezdeti értékadása
	String	String kezdeti értékadása
	Num.	Numerikus kezdeti értékadás
	Nil	Nil érték írása változóba (kezdeti értékadás)
	Kódbl.	Kódblokk kezdeti értékadása
%	Num.	Modulószámítás
+=	Dátum	Naphozzáadás és értékadás
	String	Összekapcsolás és értékadás
	Num.	Összeadás és értékadás
–=	Dátum	Naplevonás és értékadás
	String	Összekapcsolás szóközlevágással és értékadás
	Num.	Kivonás és értékadás
*=	Num.	Szorzás és értékadás
/=	Num.	Osztás és értékadás
^=,**=	Num.	Hatványozás és értékadás
%=	Num.	Modulószámítás és értékadás
$	String	Részstringkeresés

A deklarált, értékkel fel nem töltött PUBLIC változók .f. (FALSE) értéket tartalmaznak. A deklarált, értékkel fel nem töltött tömbök minden eleme NIL értékű lesz.

Ajánljuk figyelmébe!

Bármely típusú tömb dimenziónként legföljebb 4096 elemet tartalmazhat, de a dimenziók számának csak a rendelkezésre álló memória szab határt.

Az egyes változóknak deklarálásukkor kezdőértéket adhatunk. Erre szolgál az ún. kezdőértékadó operátor (:=). Tömböknek is adhatunk így kezdőértéket, ekkor az egyes elemek értékét vesszővel elválasztva kell felsorolni, és az egész listát kapcsos zárójelek közé kell tenni.

A Clipper programokban, mint minden más adatbázis-kezelő rendszerben fontos dolog az adatbázis adatainak és a memóriaváltozóknak a megkülönböztetése. Ez igazán csak akkor érdemel figyelmet, ha egy memóriaváltozó és egy adatbázismező azonos néven szerepel. A Clipper nyelvben az a szabály, hogy a változó előtt szereplő előtag szabja meg, miszerint az adott helyen a memóriaváltozót vagy az adatbázismezőt kell használni. A FIELD-> előtag adatbázismező alkalmazását írja elő, a MEMVAR-> előtag pedig a memóriaváltozóét. A két előtag használatát a fordítóprogramnak megadott opcionális kapcsolókkal is befolyásolhatjuk, de a legtisztább megoldás mégis az explicit megadásuk.

Ha valamelyik változóra többször hivatkozunk, nem szükséges minden egyes esetben az előtagot alkalmazni: elég, ha az eljárás elején, minden végrehajtható utasítást megelőzően MEMVAR vagy FIELD kulcsszóval meghatározzuk, hogy az adott azonosító memóriaváltozót vagy adatbázismezőt reprezentál. Jobb azonban vigyáznunk arra, hogy adatmező és memóriváltozó ne szerepeljen egyforma azonosítóval; így az összes ebből adódó kellemetlenséget és hibalehetőséget elkerüljük.

Az új Clipper-verzió az újonnan bevezetett tárolási osztályok mellett néhány különleges változótípust is kínál. Az előző változatokból is ismert DÁTUM, KARAKTER, LOGIKAI, MEMO, NUMERIKUS, TÖMB típusok mellé most három új típus került: a KÓDBLOKK, a NIL és az OBJEKTUM.

A NIL mint adattípus egy érdekes kreáció. A NIL típusú változó egyetlen lehetséges értéke a NIL. Ha egy változó NIL típusú (egyenlő NIL értékkel), akkor ez azt jelenti, hogy a változó már létezik, de használható információt nem tartalmaz. NIL értéket bármilyen más típusú változó felvehet, és ezzel rögtön NIL típusúvá is válik.

A KÓDBLOKK tulajdonképpen egy olyan függvény, amely nem tartalmazhat deklarációs és ciklusutasításokat (ez nem jelenti azt, hogy a KÓDBLOKK-ból hívott függvény sem tartalmazhat ilyeneket). A KÓDBLOKK-ot reprezentáló változóban a Clipper a blokk kódrészének a címét tárolja. Ezt a kódcímet tartalmazó változót egy KÓDBLOKK-futtató függvénynek adhatjuk át, amely végrehajtja a változó által jelzett címen elhelyezkedő kódot. A futtató függvényeken keresztül paramétereket is ad hatunk át a KÓDBLOKK-nak. A KÓDBLOKK utasításrésze több, egymástól vesszővel elválasztott utasítást is tartalmazhat. Visszatérési értéke a kódrész utolsó kifejezésének értékével egyenlő.

A harmadik új változótípus az OBJEKTUM. Mivel a sorozat első négy részében másról nem is nagyon volt szó, mint a Clipper objektumairól, ezekről most nincs mit mondanunk.

Bevezettek néhány új operátort is; egy részük az újdonsült változótípusokhoz kötődik, mások pedig a régi változótípusokkal végeznek új műveletet. Közöttük vannak azok, a C nyelvből ismert operátorok, amelyek egyetlen lépésben végzik el a bal oldali érték megváltoztatását és az értékadást. Ezek a += -= *= /= %= ^=. A stringműveletek száma is bővült. A műveletek részletes listáját az 1. táblázat; míg az egyes változótípusokon végrehajtható függvényeket a 2. táblázat tartalmazza.

Fridl György

Változókon végrehajtható függvények

AADD( )	Tömb	Új elem adása a tömb végére
ABS( )	Num.	Abszolút érték
ACLONE( )	Tömb	Tömb duplikálása
ACOPY( )	Tömb	Tömbelemek másolása másik tömbbe
ADEL( )	Tömb	Tömbelem törlése
AEVAL( )	Kódbl.	Kódblokk végrehajtása tömbelem-paraméterekkel
AFILL( )	Tömb	Tömb elemeinek feltöltése azonos értékkel
AINS( )	Tömb	Nil értékű elem beszúrása a tömbbe
ALLTRIM( )	String	Bevezető és záró szóközök levágása
ARRAY( )	Tömb	Inicializálatlan elemekből álló tömb létrehozása
ASC( )	String	ASCII-kód meghatározása
ASCAN( )	Tömb	Keresés a tömbben
ASIZE( )	Tömb	Tömb méretének módosítása
ASORT( )	Tömb	Tömb rendezése
AT( )	String	Részstring helyének meghatározása
CDOW( )	Dátum	A nap neve
CHR( )	Num.	Szám karakterré konvertálása (ASCII)
CMONTH( )	Dátum	A hónap neve
CTOD( )	String	Dátumtípusra konvertálás
DAY( )	Dátum	A nap sorszáma (0-31)
DBEVAL( )	Kódbl.	Kódblokk végrehajtása adatbázismező-paraméterekkel
DESCEND( )	Dátum	Dátum numerikus alakjának komplemense
	Num.	Komplemensképzés
	String	String komplemensképzése
DIRECTORY( )	Tömb	Directory-bejegyzések beolvasása egy tömbbe
DOW( )	Dátum	A nap sorszáma (0-7)
DTOC( )	Dátum	Dátum konvertálása stringgé, elemelválasztókkal
DTOS( )	Dátum	Dátum konvertálása stringgé, elemelválasztás nélkül
EMPTY( )	Dátum	Változó ürességének vizsgálata
	Nil	A Nil változó mindig üres
	Num.	Változó ürességének vizsgálata
	String	Változó ürességének vizsgálata
EVAL( )	Kódbl.	Kódblokk végrehajtása
EXP( )	Num.	Természetes alapú hatványozás
HARDCR( )	*String	Lágy kocsivissza (141) jelek cseréje keményre (13)
INT( )	Num.	Törtrész levágása
ISALPHA( )	String	String első karaktere alfanumerikus?
ISDIGIT( )	String	String első karaktere numerikus?
ISLOWER( )	String	String első karaktere kisbetű?
ISUPPER( )	String	String első karaktere nagybetű?
LEFT( )	String	Bal oldali karakterek kivágása
LEN( )	String	String hosszának meghatározása
LOG( )	Num.	Természetes alapú logaritmus
LOWER( )	String	String kisbetűsre konvertálása
LTRIM( )	String	Bevezető szóközök eltávolítása
MAX( )	Num.	Két számból a nagyobb kiválasztása
MEMOEDIT( )	*String	String szerkesztése
MEMOLINE( )	*String	String egy sorának alászínezése
MEMOREAD( )	*String	String olvasása szövegfájlból változóba
MEMOTRAN( )	*String	Kocsivissza jelek kicserélése
MEMOWRIT( )	*String	String szövegfájlba írása
MIN( )	Num.	Két számból a kisebb kiválasztása
MLCOUNT( )	*String	Stringsorok számának meghatározása
MLPOS( )	*String	A sor kezdőpozíciójának meghatározása
MONTH( )	Dátum	A hónap sorszáma
PADC( )	String	String középre igazítása
PADL( )	String	String balra igazítása
PADR( )	String	String jobbra igazítása
RAT( )	String	Részstring helyének meghatározása
REPLACE	Dátum	Dátummezőbe dátumérték beírása
	Num.	Numerikus mezőbe érték írása
	*String	Stringmezőbe stringérték írása
REPLICATE( )	String	String ismétlése
RIGHT( )	String	Jobb oldali karakterek kivágása
ROUND( )	Num.	Egészre kerekítés
RTRIM( )	String	Záró szóközök levágása
SOUNDEX( )	String	Hangzás szerinti számmá konvertálás
SPACE( )	String	Szóközstring létrehozása
SQRT( )	Num.	Négyzetgyökvonás
STR( )	Num.	Szám stringgé konvertálása
STRTRAN( )	String	Részstring helyettesítése
STUFF( )	String	Részstring helyettesítése
SUBSTR( )	String	Részstring kivágása
TRANSFORM( )	Dátum	Konvertálás formázott stringgé
	Num.	Konvertálás formázott stringgé
	String	Konvertálás formázott stringgé
TYPE( )	Dátum	Változótípus meghatározása
	Nil	Változótípus meghatározása
	Num.	Változótípus meghatározása
	String	Változótípus meghatározása
UPPER( )	String	String nagybetűsre konvertálása
VAL( )	String	String numerikussá konvertálása
VALTYPE( )	Dátum	Változótípus meghatározása
	Nil	Változótípus meghatározása
	Num.	Változótípus meghatározása
	String	Változótípus meghatározása
YEAR( )	Dátum	Az évszám (mindig négyjegyű)

A stringműveletek egytől egyig végrehajthatók memo típusú változókon is.
A *-gal jelölt stringfüggvények tipikusan memoműveletek.

Könnyű-e játszani a szavakkal?

Visszatekintés

Az Alaplap új színfoltjaként 1991. októberi számunk „Visszacsatolás" rovatában jelent meg először a Kaleidoszkóp. Szándékunkat mintegy megtréfálva sajnos éppen a visszacsatolás bizonyult a legnehezebbnek, ami pedig minden rejtvényrovatnak éltető eleme. A lap előállítása (nem kis részben a lemezmelléklet „macerás" műveletei miatt) viszonylag hosszú átfutású, ezért amikor lapunk megjelenik, a következő havi szám anyagát már nyomdába kell küldeni. Nem tudjuk tehát bevárni a megfejtéseket, és nem tudunk azonnal reagálni olvasóink leveleire, megfejtéseire, pedig az volna az ideális.

Októberi feladványunk (januárig meghosszabbított határidővel) sokakat megmozgatott. Leghamarabb hozta a posta Csurgay Péter tömör, frappánsan megfogalmazott megoldását. Mint bevezető szavaiban írta: „a feladat annyira tetszett, hogy rögtön nekiláttam a megoldásnak".

Megfejtőink közül többen kiemelték, hol kellett „nem szigorúan matematikus következtetést" alkalmazni (Gyeszat Zoltán szavai), hol volt szükség „olyan ismeretre, amivel egy program nem rendelkezhet (Vágó-Láng páros), illetve hol volt az a pont, ahol „ha számítógép volnánk, itt el is akadnánk a megoldásban" (Dezső András). Ettől eltekintve is mindhármójuk megfejtésében a heurisztikus meggondolások lépéseinek különösen jól átgondolt, példaszerű leírását találtuk. Külön ki kell emelnünk Gyeszat Zoltán megfejtését, aki a legprecízebb indoklást adta, és helyenként a formális nyelvek tulajdonságaira utaló rövid megjegyzéseivel akaratlanul is elárulta, hogy azon a területen is elég otthonosan mozog.

A probléma számítógépesítéséről már erősen megoszlottak a vélemények. Volt, aki egyenesen úgy fogalmazott, hogy „ez az «ágyúval verébre» tipikus esete lenne" — és a fenti nem formális (ezért előre nem is programozható) megfontolásokra hivatkozott. Valóban, a végiggondolandó lehetőségek kis száma miatt igazat kell adnunk olvasóink ilyetén véleményének, hozzáfűzve azonban, hogy nagyobb korpusz esetén (pl. egy értelmező szótár szóanyagának feldolgozásakor) egyáltalában nem kárbaveszett fáradság a nem teljesen formalizálható nyelvi anyagnak akár többlépcsős, emberi munkafolyamatokkal tarkított számítógépes feldolgozása. A számítógép ezekben az esetekben nagyszerűen segítheti az ember intuitív munkahipotéziseinek keletkezését és ellenőrzését. Az ember-gép együttműködésnek éppen ezen a területen már kitűnő példái vannak. Szinte jegyzőkönyvi pontossággal megörökíthető a gondolatok születése, pontról pontra nyomon követhető, ahogy fokozatosan kialakul egy nyelvi adatbázisból valami minőségileg magasabb rendű dolog: az igazi tudásbázis. Nem túlzás azt állítani, hogy a számítógépes nyelvészet és a mesterséges intelligencia egymásra találásának jelenleg ez az egyik perspektivikus területe.

További kalandozás a nyelvek és a számok birodalmában

Van egy egészen különleges nyelvcsalád a világon, amelynek egyes nyelveit ezer kilométerek választják el egymástól. E nyelvek csodálatos módon majdnem változatlanul maradtak több mint fél évezreden keresztül, és ma is híven őrzik a nyelv akkori állapotát. Ez annál is inkább érdekes, mert a rokon nyelvek összehasonlításából jól következtethetünk arra, hogyan különültek el, és milyen sorrendben váltak le újabb és újabb csoportok a közös törzsről. Számítógépes nyelvstatisztikai feldolgozások alapján egyes kutatók arra is készítettek becsléseket, hogy mikorra tehető a szétválások időpontja. Archaikus voltuk ellenére az egyes rokon nyelvek természetesen változtak, ám ezek a változások a legtöbb esetben szokatlanul szigorú hangtani következetességgel mentek végbe.

Sok egyéb érdekes tulajdonságuk mellett ezek a nyelvek azzal tűnnek ki a világon ismert sok ezer nyelv közül, hogy ők használnak a legkevesebb hangot: mindössze 10 mássalhangzóra és 5 magánhangzóra van szükségük mondanivalójuk kifejezésére. Egyikük pedig összesen csak 13 hangot használ.

A tudósok rekonstruálták e nyelv számneveinek eredeti alakját. Ezek közül mutatunk be néhányat, éppen csak annyit, amennyi elegendőnek látszik olvasóink elképesztésére. Kérésünk lényegében csak annyi, hogy e rövid szójegyzékből kisajtolható információk segítségével próbáljanak meg következtetni, és önállóan felírni néhány további számnevet ezen az archaikus nyelven.

Íme, az ismertnek feltételezett számnevek jegyzéke:

2 = lua
3 = kolu
5 = lima
6 = ono
7 = hiku
11 = umi-kumana-kahi
49 = iako me ka iwa
57 = iako me ka umi-kumana-hiku
490 = lau me ka lua iako me ka umi
5000 = mano me ka lua lau me ka lima iako

Aki a fenti szójegyzéket alaposan áttanulmányozza, szabályszerűségeit kibogozza, annak már gyerekjáték lesz (?) megfelelni első kérdésünkre:

1. Hogyan mondják ezen a nyelven a következő számneveket: 4845, 286, 89, 41, 17?

További kérdéseink inkább csak tájékozódó jellegűek:

2. Mit gondolnak, melyik nyelvcsaládról lehet szó?

3. Milyen messze lehet Afrikától e nyelvcsalád egyik közeli rokona?

4. Van-e valamilyen hipotézise, hogy mi lehet e különös számrendszer kialakulásának magyarázata?

Az értékelésben létraversenyszerűen összesítjük a szerzett pontokat. Az „éllovasokat" félévenként az Alaplap, illetve a Floppy.Lap éves előfizetésével díjazzuk. Az első díjkiosztás időpontja: 1992. április 30, amikor is az első három helyezett számára egész éves előfizetést biztosítunk az Alaplapra és a Floppy.lapra is. Az utánuk következő 5 helyezett számára a Cédrus Karolina Áruház külön jutalmakat ad.

A létraverseny állása 1992. január 11-én: Dezső András, Gyeszat Zoltán (90–90), Láng Attila D. (70), Süle Gábor (40), Csurgay Péter, Nagy Zoltán, Vágó Dániel (30—30).

A megfejtéseket a szerkesztőség címére (1441 Budapest VIII., Reguly Antal u. 8.) kérjük beküldeni 1992. március 10-ig. A féléves ciklus lezárására való tekintettel kivételesen lehetővé tesszük, hogy olvasóink március 10-ig pótlólag még beküldhessék a januári feladatok megfejtését is.

Vargha Dénes

SZÁMÍTÁSTECHNIKA KULCSRAKÉSZEN!

A LEGKISEBB NOTEBOOK-TÓL A LEGGYORSABB 486-OSIG

- XT, AT, 386, 386SX, 486, Laptop minden kiépítésben.
- EPSON, STAR, NEC nyomtatók teljes választéka.
- MODEMEK és egyéb tartozékok széles választéka.
- Magánszemélyeknek KÉSZPÉNZFIZETÉS ESETÉN KEDVEZMÉNY!
- ASHTON-TATE, BORLAND, MICROSOFT, NANTUCKET, LOTUS szoftverek.
- SHAREWARE programok (1200-féle) 360,– Ft + ÁFA áron.
- MODEMES táv-adatátviteli és BBS rendszerek szállítása.
- VÍRUSÖLŐ program (120-féle vírust öl!)
- NOVELL HÁLÓZATI SZOFTVEREK, hálózatkiépítés.

Ajánlatunk:
NOTEBOOK SZÁMÍTÓGÉPEK MÁR 69 900,– Ft-tól

AT számítógép: 1 MB RAM, 40 MB HDD, 1,2 MB FDD, Mono 14" (PHILIPS) 1 S, 1 P, 101 gombos bill. 63 100,– Ft + ÁFA
(Készpénzért 59 900,– Ft + ÁFA.)

Amikor ezt a hirdetést Ön olvassa, áraink már úgyis alacsonyabbak! Ezért kérjük, telefonáljon vagy írjon, és mi örömmel adunk felvilágosítást, küldünk részletes árjegyzéket!

QWERTY

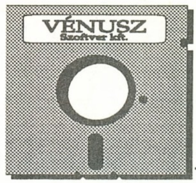
High Tech. Kft.
1117 Budapest XI., Orlay u. 4.
Telefon: 166-3098, 185-2687, Fax: 185-2687
BBS: 118-7950 BUDAPEST BBS

INFORMÁCIÓKÉRÉS: 13 ▶

VÉNUSZ
kiváló áruk fóruma

Általános nyilvántartó és kalkulátorprogram

Egy igazi alkalmazásgenerátor, amellyel programozói ismeretek nélkül bármilyen nyilvántartási rendszer elkészíthető!



1145 Bp. XIV., Amerikai út 39.
Tel.: 183-0722 Angyal József
183-0720 Angyal Judit

INFORMÁCIÓKÉRÉS: 17 ◀

- AT 286/386/486 igény szerinti kiépítésben
- EPSON nyomtatók és kiegészítők teljes választéka
- Hardveralkatrészek nagy választékban
- Hálózattervezés, -építés
- Átalánydíjas és eseti szerviz

Készpénzes ajánlatunk:
Alapkonfiguráció: alaplap, RAM, 1,2 Mbájt FDD, 40 Mbájt HDD, IDE, multi I/O, MGP, 14" mono monitor, Baby-ház, 101 gombos billentyűzet.

AT 286-12/16 + 1 Mbájt	58 800,–
AT 286-16/21 + 1 Mbájt	60 800,–
AT 286-20/25 + 1 Mbájt	74 800,–
AT 386-25/32 + 2 Mbájt	103 100,–
AT 386-33/54 + 64 kB Cache + 2 Mbájt	122 000,–
AT 386-40/65 + 64 kB Cache + 2 Mbájt	139 000,–
ISA 486-25/114 + 64 kB Cache + 4 Mbájt	190 400,–
ISA 486-33/150 + 256 kB Cache + 8 Mbájt	232 400,–

Kérje részletes árlistánkat!

Áraink 12 hónap cseregaranciával, ÁFA nélkül értendők.

UNITRADE
Szervezési, kereskedelmi és Számítástechnikai
K.F.T.

1073 Budapest VII., Erzsébet krt. 48.
Telefon/Fax: 142-2115

...nem csak számítástechnika

INFORMÁCIÓKÉRÉS: 15 ▶

Barex
Computer Kft.

SZAKÜZLETEK:

Bp. V., BAJCSY-ZSILINSZKY ÚT 54.
TEL./FAX: 111-6025
Bp. V., BAJCSY-ZSILINSZKY ÚT 64.
TEL./FAX: 131-1960
A KÍNÁLATBÓL:

Monitorszűrő (üveg)	4 000,–
Egér (Microsoft comp.)	1 400,–
Egéralátét	160,–
A/4 (OASCAN HF9100)	72 000,–
Mono monitor 14", papírfehér	7 800,–
Qmouse (2050 dpi, dinamikus)	3 200,–

QUANTUM, CONNER WINCHESTEREK NAGY VÁLASZTÉKBAN!

INFORMÁCIÓKÉRÉS: 23 ◀

Belépőjegy a CeBIT-re

Elővételi áron,
forintért megvásárolhatja belépőjegyét
a világ legnagyobb
számítástechnikai szakvásárára,
a Hannoverben idén március 11-18 között
megrendezésre kerülő CeBIT-re.

Napi jegy:	1055,– Ft
Bérlet:	2610,– Ft
Katalógus:	1505,– Ft

Presentex Vásárképviseleti Kft.
Vásárközpont,
Budapest X., Albertirsai út 10.
Telefon: 157-4280, 178-0352 Fax: 163-2605

INFORMÁCIÓKÉRÉS: 29 ▲

Alapvetően **ÚJ!** koncepció

LAN-GUARD

Integrált hálózati biztonsági rendszer

Lokális hálózatok
VÍRUS- és ADATVÉDELME

- FILE-SERVER-ek, terminálok hozzáférés- és bootvírus-védelme.
- File-vírusok elleni védelem.
- Integrált munkafolyamat-vezérlés.
- Egyedi számítógépek védelme.

Ha fontosak az adatai, segít a

DATA DOCTOR Kft.
1149 Budapest, Buzogány utca 4.
Telefon/Fax: 18-37-299

INFORMÁCIÓKÉRÉS: 22 ▲

FAN computer

KIVÁLÓ MINŐSÉGŰ SZÁMÍTÓGÉPEK 24 HÓNAP GARANCIÁVAL!

SecureData

MEMÓRIAKÁRTYÁS SZUPERBIZTONSÁGOS ADATVÉDELMI RENDSZEREK

FELLOW
KÖNYVMÉRETŰ ASZTALI SZÁMÍTÓGÉPEK

RÉSZEGYSÉGEK, „QUANTUM" WINCHESTEREK, MOUSE-OK, SCANNEREK, DIGITALIZÁLÓ TÁBLÁK

FAN Electronics Ltd

Tajvani—Magyar Vegyes vállalat
1118 Budapest, Késmárki u. 6.
(volt Friss István u.) Telefon/Fax: 185-0813

INFORMÁCIÓKÉRÉS: 09 ▲

MACRODA – A MODERN SZÁMÍTÁSTECHNIKA!

„THE MACRO" számítógépek
1+2 év garanciával,
CAD rendszerek,
3 M mágneses adathordozók,
mágneskártyás adatvédelmi rendszerek,
számítástechnikai kiegészítők,
STAR és **CITIZEN** nyomtatók,
CANON irodatechnika.

Kérje részletes árlistánkat!

MACRODA®

MACRODA KFT.

MINTABOLT:	KERESKEDELMI IRODA:
1123 Bp., Alkotás u. 21.	1016 Bp., Szirtes u. 28./A
Tel./Fax: 156-4802	Tel.: 186-5782, 186-5686
	Fax: 186-5686, Telex: 22-5375

INFORMÁCIÓKÉRÉS: 08 ▲

Végtelenül bővíthető

Palettánkon most egy olyan számítógépcsalád szerepel, amelyet „látásból" valamennyien jól ismerünk, hiszen plakátokon, újsághirdetésekben megragadják a szemünket. Kíváncsiak voltunk, mit is tud a nagy „csinnadrattával" beharangozott gépcsalád.

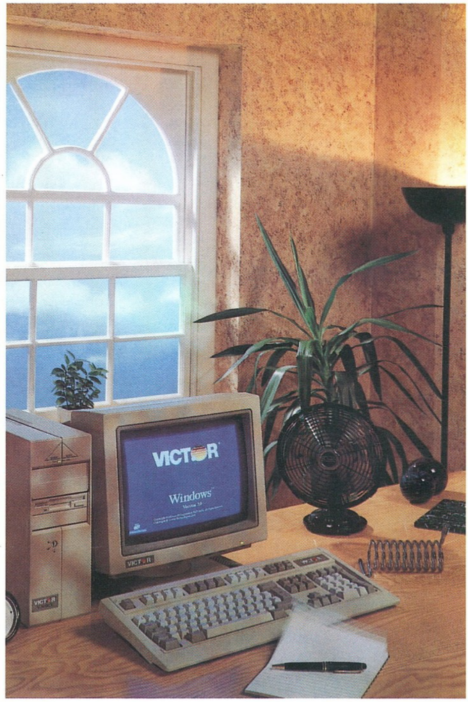
A svéd tervezésű Victor-család október közepén jelent meg a hazai számítástechnikai piacon. Repertoárja a 286-os processzorú gépektől felfelé terjed. Valamennyi merevlemezes készülékhez hivatalosan installálják az MS-DOS 5.0 és a Windows 3.0 operációs rendszereket. A Windows mellett a HunWin is „tartozék", így Victor a magyar ékezetes karaktereket is kezeli, és gravírozott magyar klaviatúrával forgalmazzák. A gépeket többnyire VGA monitorral értékesítik, és a 386 SX gépektől felfelé Microsoft egér is jár valamennyi Victorhoz.

A gépcsalád jellemzője az a cserélhető CPU-kártya, amely egyúttal a grafikus részt is tartalmazza. Ez a megoldás jelentősen gyorsítja a gép működését, ami grafikus környezetben (Windows alatt) nagyon fontos. Költségkímélő megoldás továbbá, hogy egy 286-osról 386-osra való átálláskor — amely manapság elég gyakori igény — elég csak a CPU-kártyát kicserélni, így az átállás nem is kerül olyan sokba.

Minden Victorhoz a fix merevlemezen (mely 52 Mbájtostól 420 Mbájtos lehet) kívül egy „kivehető" harddiszk (ADD-PAK) is tartozik. Ez teszi lehetővé a kapacitás bővítését. Az ADD-PAK kapaciástartománya megegyezik a fix diszkével. Így első megközelítésben 840 Mbájtig bővíthető a gép. Mivel azonban az ADD PAK utolsó, max. 420 Mbájtja „kivehető", így azt egy másikra cserélve, újabb 420 Mbájttal bővíthető a gép. Ezzel a „láncolással" elvileg a végtelenségig növelhető a Victor kapacitása.

Éppen az ADD-PAK alkalmazása teszi rendkívül biztonságossá a gépet. Nem véletlen, hogy a nagy adatbiztonság miatt elsősorban kormányhivatalokban, bankokban és a hadseregben használják előszeretettel a Victort. Ugyanis minden olyan információt, amit félteni kell, ADD-PAK-en tárolnak. Kulcs is tartozik a gépekhez. Az ADD-PAK-et csak az veheti ki, akinél a kulcs van, s éjszaka széfben tárolják az ADD-PAK-et. Reggel pedig csak az kaphatja meg, aki „bejáratos" a széfbe. A gépben pedig megint csak a „kulcsos ember" férhet a kényes adatokhoz. A Victor védelmet nyújt az ADD-PAK „kitépése" ellen is. Ilyenkor megáll, és a gép 2-3 percig használhatatlan. A gépcsalád valamennyi tagjánál az ADD-PAK mérete azonos, a legkisebbtől a legnagyobb gépig „adalék" lehet ugyanaz a hordozható ADD-PAK.

A rendkívül csendes (max. 33 dB) Victor nem tartozik a drága gépek közé. Például egy Victor 386MX — amely a leginkább kedvelt típus — 220 000 forint körüli árával már sokak számára hozzáférhető. Kis helyfoglalásával — felcsavarozható az íróasztallap alá is! — szűkös irodákban is jól megfér. A hálózatban (Novell alatt) is működő gépből — a nagy adatbiztonság itt ugyancsak vonzó! — már szép számmal adtak el. Ilyenkor a gép mint munkaállomás csupán egy monitorból és egy klaviatúrából áll, sem fix diszk, sem pedig floppymeghajtó nem kell hozzá.



VICTOR® TECHNOLOGIES.

A Victor-felhasználók kiszolgálását növeli az a hotline-támogatás, amellyel segítenek a „bajba jutott" felhasználókon. Telefonon keresztül a szakember el tudja dönteni, hogy szoftver vagy hardver eredetű problémával szembesült-e a felhasználó. A szoftvert érintő nehézségek orvoslásához általában elég a „távsegítség". A hardverből fakadó gondokon pedig a gépek egy újabb „tartozéka", egy ellenőrző program segít. Hardverhibánál „betájolja" a felhasználót a hiba forrását illetően. Így a hibák nagy része akár telefonon keresztül is javítható, s csak ritkán fordul elő tényleges szervizes beavatkozás.

Sziebig Andrea

Starcz Andor:
DataFlex tippek és trükkök

PAN–TYPE KKT., 1991 Pécs, 256 oldal, 2990 Ft.

Az első magyar nyelvű könyv, mely alkalmazási feladatokat tárgyal, van floppylemez melléklete, és a több mint 120 operációs rendszer alatt működő DataFlex relációs adatbázis-kezelő környezetében fejlesztőknek szól. Igényes kiadvány.

A szerző öt fejezeten keresztül hatékony, új parancsokat kínál, melyek mindegyike a floppylemezen található — alkönyvtárakra bontva. Az első fejezet tulajdonképpen szervezői információgyűjtemény. (A rendszer, projekt, modell, szoftverminősítés.) A második fejezet a DataFlexre vonatkozó alapinformációkat tartalmazza — főként a window/image paraméterekre —, jól értelmezhető táblázatba szerkesztve. Harmadik fejezete a képernyővezérlések gyakorlati alkalmazására ad példákat. Külön kiemelendő a szerző által „multiwindow"–nak nevezett képtechnika, mely a 2.3b verzióval bezárólag eddig külön programozást igényelt. (Tetszőleges képen kurzorsor mozgatása!) A funkció ezentúl egy paranccsal aktiválható! A negyedik fejezetben „ablakot" nyit bármelyik DataFlex adatbázisra, s annak adatai tetszőleges nevű képen görgethetők.

Érdekes része a könyvnek az akár merevlemeznyi tömbök használatának módszere, melyre támaszkodva egyszerűbb a regressziós egyenes illesztése — korrelációs együttható számításával. A módszer integrálja az előzőekben felsoroltakat. Kár, hogy a számítás matematikai indoklását a könyv nem tartalmazza! Végül a könyv a gyakorló rendszerfejlesztőknek kínál egy komplex fejlesztői keretvázat, valamint egy komplett jogosultsági rendszer programjait.

A lemezmellékletet csak a DataFlex futtató- és fejlesztőrendszerével rendelkezők használhatják, de erre sajnos nem figyelmezteti a járatlan olvasót!

A lemezmelléklet minden programja hibamentesen futott mind egyedi mind többfelhasználós környezetben, mind DOS 3.xx, mind Novell-hálózaton, míg Unix, Xenix környezetben a terminálokat fel kellett készíteni a 127 bájt feletti IBM-karakterkészlet megjelenítésére. Ajánlható a könyv mindenkinek, aki rendszerfejlesztéssel foglalkozik, és főként mindazoknak, akik erre a DataFlex fejlesztői rendszert használják. Nekik minden bizonnyal megtérül a könyv reálisan borsos ára.

ICS
Identcode-System

Nekünk 0,2 másodpercre volt szükségünk ennek a vonalkódnak a kinyomtatásához. Ez Önnek is sikerülhet. Az ICS–PZ etikettnyomtató család bármely tagjának segítségével.

Mag ICS
Informatikai Rendszerfejlesztő és Marketing Kft.

H-9400 Sopron, Bástya u. 75.,
Tel.: ++36-99-14 250, ++36-99-34 035
Fax.: ++36-99-14 250

Budapesti Képviselet:
1111 Bp., Lágymányosi u. 14.
Tel./Fax: ++36-11-650 272

Bibliográfia

Összeállításunkban a hónap témájához — a programnyelvek világához — kapcsolódó könyvek között válogattunk.

Adorján Noémi: FORTH lépésről lépésre.
Budapest, 1990. Műszaki Könyvkiadó, 202 oldal.

A Logo programozási nyelv.
Budapest, 1986. Műszaki Könyvkiadó, 317 oldal.

Bakos Tamás: A COBOL programozási nyelv.
Budapest, 1974. Műszaki Könyvkiadó, 344 oldal.

Germain, C. B.: IBM PC XT/AT programozói kézikönyv. (BASIC, COBOL, Pascal, FORTRAN és Assembler ismertetésével)
Budapest, 1988. Novotrade, 388 oldal.

Horowitz, Ellis: Magasszintű programnyelvek.
Budapest, 1987. Műszaki Könyvkiadó, 334 oldal.

Jensen, Kathleen — Wirth, Niklaus: A PASCAL programozási nyelv. Felhasználói kézikönyv és a nyelv formális leírása. (2., bővített kiadás)
Budapest, 1988. Műszaki Könyvkiadó, 225 oldal.

Kaufman, R. E.: FORTRAN képeskönyv.
Budapest, 1983. Műszaki Könyvkiadó, 285 oldal.

Kernighan, B. W. — Ritchie, D. M.: A C programozási nyelv.
Budapest, 1985. Műszaki Könyvkiadó, 231 oldal.

Lipovszki György — Subai László — Beszeda Tamás: FORTH programozási rendszer és nyelv.
Budapest, 1985. LSI ATSZ, 245 oldal.

Lőcs Gyula: Az Algol 60 programozási nyelv.
Budapest, 1971. Műszaki Könyvkiadó, 254 oldal.

Lőcs Gyula — Sarkadi Nagy István — Szlankó János: A BASIC programozási nyelv.
Budapest, 1976. Műszaki Könyvkiadó, 290 oldal.

Lőcs Gyula — Vigassy József: A FORTRAN programozási nyelv. (6., bővített kiadás)
Budapest, 1985. Műszaki Könyvkiadó, 336 oldal.

Márkusz Zsuzsanna: PROLOG-ban programozni könnyű.
Budapest, 1988. Novotrade Rt., 227 oldal.

Nagy Kálmán: Strukturált programozás COBOL nyelven.
Budapest, 1980. SZÁMALK, 343 oldal.

Nestle, Fritz — Ostertag, Eberhard: BASIC, LOGO, Pascal.
Budapest, 1988. Novotrade, 149 oldal.

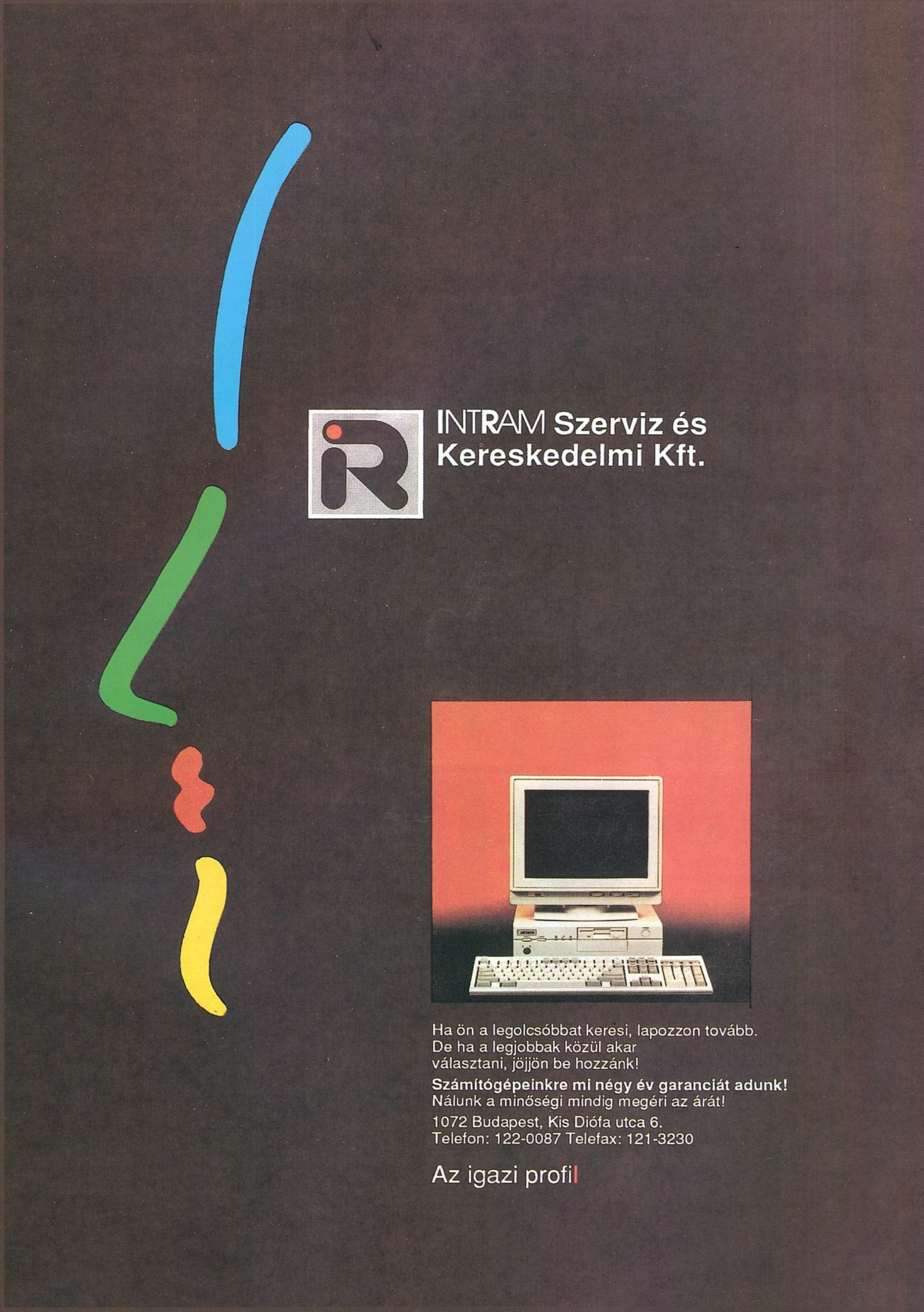
Pyle, I. C.: Az Ada programozási nyelv.
Budapest, 1987. Műszaki Könyvkiadó, 310 oldal.

Rákosi Miklós: A PL/1 programozási nyelv.
Budapest, 1978. Műszaki Könyvkiadó, 502 oldal.

Seprődi László: A GPSS szimulációs nyelv.
Budapest, 1980. Műszaki Könyvkiadó, 302 oldal.

Traister, R. J.: BASIC-ből a C-be.
Budapest, 1988. Prentice Hall — Novotrade Rt., 138 oldal.

Zimányi Magdolna — Kálmán László — Fadgyas Tibor: A LISP programozási nyelv.
Budapest, 1989. Műszaki Könyvkiadó, 428 oldal.



INFORMÁCIÓKÉRÉS: 06 ▶

TELEFAX

ÍRÓGÉP

RANK XEROX

HIVATALOS DEALER ÉS MÁRKASZERVIZ

PARTNER AZ IGÉNYESSÉGBEN!

NTT-2000

Trade and Service Ltd.

MÁSOLÓGÉP

LÉZERNYOMTATÓ

1103 Budapest X., Gyömrői út 86.
Telefon: 147-2734, 147-2735
Telefax: 147-2301

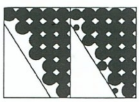
INFORMÁCIÓKÉRÉS: 04 ◄

® CONTROLL Rt

Jobb, ha éles



A számítástechnikában az idő valóban pénz, a meghibásodott gép miatt elveszett idő pedig kidobott pénz.
A *CONTROLL* által forgalmazott **Hewlett-Packard PC**-k a legmegbízhatóbbak közé tartoznak, így megkímélik Önt a veszteségektől.



HEWLETT PACKARD
Authorized Dealer

CONTROLL - EGYETLEN A SOK KÖZÖTT

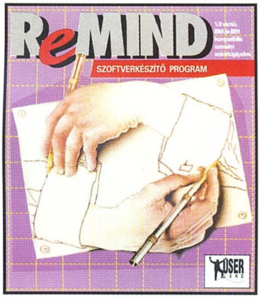
CONTROLL ELEKTRONIKAI ÉS SZÁMÍTÁSTECHNIKAI RÉSZVÉNYTÁRSASÁG
1091 Budapest, Üllői út 101. Telefon: 133-5960, 134-3324, 114-0211, 113-6243
Telex: 20-2535 Telefax: (36)-1-133-7392
Bemutatóterem: Budapest IX.,Üllői út 101.

- KÖVESDI DESIGN -

INFORMÁCIÓKÉRÉS: 05 ◄

ReMIND





A REMIND nemcsak egy új szoftver, hanem egyben egy új technológia, mellyel gyorsabban és olcsóbban lehet jó minőségű felhasználói programokat készíteni, mint a jelenlegi negyedik generációs szoftverekkel.
A REMIND nem pusztán csak egy új szoftver hanem az adatfeldolgozás jövője is.
A REMIND kezelése egyszerű, gyorsan megtanulható,
segítségével a szoftver készítésére fordított idő a töredékére is csökkenhet.
A legegyszerűbb feladatoktól a legigényesebb programokig minden PC alkalmazónak időt és energiát takarít meg.

ReMIND -A LEGRÖVIDEBB ÚT.

USER LAND

1121 Budapest, Konkoly Thege Miklós út. 19. B/C Tel.: 1695-140, 1695-449

INFORMÁCIÓKÉRÉS: 02 ▼



INFORMÁCIÓKÉRÉS: 03 ▲